Examination scheduling using the ant system.

Pugh, Nicholas John

Examination Scheduling Using The Ant System

by

Nicholas John Pugh
BSc (Hons) (University of Wales, Swansea), MSc (Lancaster)

Thesis

submitted to the University of Wales
in candidature for the degree of

PHILOSOPHIÆ DOCTOR

European Business Management School
University of Wales Swansea
Swansea SA2 8PP
United Kingdom

March 2004

Declaration

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

NICHOLAS JOHN PUGH
1 March 2004

Statement

This thesis is the result of my own investigation, except where acknowledgement of other sources is given.

NICHOLAS JOHN PUGH
1 March 2004

Statement

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan (subject to the law of copyright), and for the title and summary to be made available to outside organisations.

NICHOLAS JOHN PUGH
1 March 2004

Acknowledgements

There are so many people that I would like to acknowledge for all their support and patience over the duration of my PhD.

First and foremost, my deepest gratitude goes to Dr. Jonathan Thompson who has provided guidance, support, motivation and friendship. Without him, the completion of this PhD would not be possible.

Many thanks must go to Prof. Ebe Bischoff who has been a great confident and editor, especially over the last 18 months and Dr. Kath Dowsland who is thanked for pointing me in the right direction during the infancy of my study and without her, I would not have even started a PhD.

I am very grateful for the EPSRC who provided me with three annual studentships and therefore, gave me the subsidence to concentrate on my studies.

Personally, my deepest loving thanks have to go my parents, John and Lynwen, who have encouraged me every step of the way during my academic career. They have always given me advice when needed and have never demanded that I get a proper job. They have supported me in more ways that are tangible and I will always be grateful for what they have done for me. Similarly, my sister, Natalie, deserves a significant mention. She is not only my sister but also a truly great friend and has made me smile on many occasions when needed. Also, I have been privileged to work alongside some fantastic people, some past and present, and I have accumulated stories that will warm my heart for years to come. Particular mention must go to Baz, Huw, Jamie and Rich who will be best friends for years to come.

No sentimentality would be right without a mention of my Louisa. You mean more to me than you know and cannot wait to be able to spend more time with you. Thanks for your continual support and patience!!!

Summary

This work is concerned with heuristic approaches to examination timetabling. It is demonstrated that a relatively new evolutionary method, the Ant System, can be the basis of a successful two-phase solution method. The first phase exploits ant feedback in order both to produce large volumes of feasible timetables and to optimise secondary objectives. The second phase acts as a repair facility where solution quality is improved further while maintaining feasibility. This is accomplished without increasing computational effort to unrealistic levels.

The work builds on an existing implementation for the graph colouring problem, the natural model for examination scheduling. It is demonstrated that by adjusting the graph model to allow the accommodation of several side constraints as well incorporating enhancement techniques within the algorithm itself, the Ant System algorithm becomes very effective at producing feasible timetables. The enhancements include a diversification function, new reward functions and trail replenishment tactics.

It is observed that the achievement of second-order objectives can be enhanced through a variety of means. A modified elitist strategy (*ERF*) significantly improves the performance of the Ant System due to the extra emphasis on second-order feedback. It is also shown that through the incorporation of the *ERF*, trail limits and, in particular, 19^{th} century evolutionary theory the area of the solution space explored by the ants during the infancy of the search can be reduced. In addition, a good level of exploration is maintained as the search matures. This balance between exploration and exploitation is the main determinant of solution quality.

The use of a repair facility, as is common practice with evolutionary algorithms, encourages fitter solutions. The interaction between Lamarckian evolution and searching in an extended neighbourhood through the graph theoretic concept of Kempe chains leads to better overall solutions.

To Mum and Dad

In Loving Memory of Gramps

Figures

Chapter Three

Chapter Four

Chapter Five

Appendices

Tables

Chapter Two

Chapter Three

Chapter Four

Chapter 1

Introduction

1.1 Background

Our lives are ruled by structure, but often we are not aware of this. Examples are the trains we catch, the television we watch and the football matches we attend. All such events have to be scheduled and we fit our lives around them. Many are organised in an attempt to please the majority. The scheduling of exams is no different. All exams tend to cause stress and among the most difficult and important in many people's lives are their degree exams. These are taken in the final, and some in the penultimate, year in university and are the culmination of several years work and are crucial in deciding the working life fate of many students. Consequently, most institutions are interested in producing timetables which are designed to include some degree of student comfort. The examination timetabling problem is known to be computationally difficult and comprises a mixture of hard and soft constraints. A successful timetable is one that satisfies all the hard constraints and deals with the other constraints as far as possible.

1.2 The Nature of the Problem

The way that the examination timetabling problem is solved tends to vary due to different objectives demanded by institutions. Due to increasing numbers of students and greater numbers of modules, producing an examination timetable has become a more difficult task to tackle. Initially, institutions used manual techniques. At the University of Wales Swansea, this style of timetable construction was performed until 1993 when no feasible (see glossary of terms for definition, Appendix 1.1) timetable could be produced in this manner. A more sophisticated approach was needed and The Integrated Scheduling System for University Examinations (TISSUE) was introduced, Thompson (1995).

Over the years, many heuristic based methods have been used to schedule examination timetables at universities. The heuristics range from greedy to tailor made. Additionally, the use of meta-heuristics (see glossary of terms for definition, Appendix 1.1) has become more commonplace. Simulated Annealing (see TISSUE above), Tabu Search and more recently Genetic Algorithms have all been used. The focus of this thesis is the development

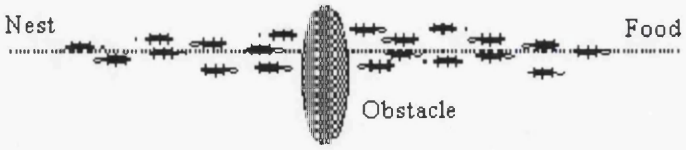
and analysis of an evolutionary meta-heuristic, called Ant Algorithms, which bases its construction style on the analogy of how real ants hunt for food. The method will be considered in greater detail later in this Introduction.

1.3 Introduction to Ants

Marco Dorigo introduced the first Ant System (AS) in his PhD thesis in 1991. AS was initially applied to the Travelling Salesman Problem (TSP), see definition in Appendix 1.2, due to the natural parallel to routing problems. The idea behind AS is (loosely) based upon the activities of the ant species *Linepithema Humile*. Tasks such as the hunt for food or the construction of nests require cooperation between the ants. Given that ants do not have sight and are unable to communicate directly, a form of stimulus is required to enable successful teamwork, namely pheromones, which act as a communication tool for the ants. To achieve greater understanding, we will consider the foraging activities of the ants. If one has ever observed ants hunt for food, it has probably been noticed that the insects tend to follow the shortest path between the nest and the food source. This can be explained through the following diagrams.

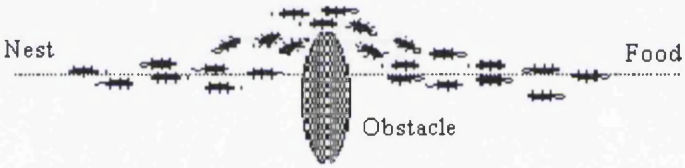
The ants follow the shortest path possible from the nest to the food source due to high deposits of pheromone.

An obstacle (pebble) is placed in the centre, in an asymmetric position.



When the ants reach the obstacle, they are unsure which path to take due to the absence of pheromone. Consequently, approximately equal numbers of ants traverse each possible path.

However, there is a quicker accumulation of pheromone per unit time on the shorter side. This greater accumulation of stimuli will bias more ants to follow the shorter path, each leaving further trail on the shorter side. Eventually, high concentrations of pheromone will be present on the shorter side and zero pheromone on the longer side. In consequence, all ants will converge to the shorter side and thus, will follow the shortest path.



The above illustrates the ability of the ants when they work collectively.

Real ants are biased towards the levels of pheromone that are present on the ground. Their foraging activities are modelled probabilistically, with the probability of the next decision dependent on pheromone, τ, and desirability, η. The latter component attempts to exploit greediness through providing knowledge of the topology of the problem. Typically, the distance between towns acts as a desirability function for TSP.

In AS, ants make decisions according to the random proportional rule (RPR). For each artificial ant in the TSP, the probability, p_{ij}, of selecting the next town j to visit when at town i at time t is computed as the product of τ and η scores, relative to the other towns j. Parameters α and β are used to balance the importance between trail and desirability. Selection is finalised through the roulette wheel philosophy.

The RPR is presented below in Equation 1.1.

$$p_{ij} = \begin{cases} \dfrac{[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta}}{\sum_{r \in J_k(i)} [\tau_{ir}]^{\alpha}.[\eta_{ir}]^{\beta}} & if\ j \in J_k(i) \\ 0\ otherwise \end{cases} \qquad (1.1)$$

where $J_k(i)$ refers to the list of towns j that can be sequenced immediately after town i when ant k is constructing a tour. The list $J_k(i)$ can depend on criteria such as nearest subset of towns, and ordering constraints i.e. towns 3 or 5 has to come after town 2.

Dorigo et al. (1991) presented three AS algorithms, Ant Density, Ant Quantity and Ant Cycle, which differed according to their trail update procedure. Ant Density ants deposit a fixed amount Q, while Ant Quantity ants deposit a measure $\frac{Q}{d_{ij}}$ (with d_{ij} reflecting the distance between towns i and j). Both methods update trail measures after the addition of each edge to the tour. Ant Cycle differs through the timing of pheromone updates. Ants replenish the trail levels on each edge (i,j) after the construction of a solution through the feedback function $\frac{Q}{L^k}$, where L^k refers to the length of tour k. All methods allow for pheromone evaporation, which aids the construction process. Relatively poor solution characteristics become less influential through time, thus reducing the chances for future selection. The combination of pheromone dissipation and reduced selection will encourage the avoidance of repeating known poor solutions. Results indicated that the Ant Cycle algorithm was superior, Dorigo et al. (1991), thus making the other algorithms obsolete and no subsequent research was performed on Ant Density and Ant Quantity.

By 1996, Ant Algorithms became known as Ant Colony Optimisation (ACO) to accommodate the emergence of extended and enhanced versions of the AS paradigm such as the MIN-MAX Ant System (MMAS) and the Ant Colony System (ACS). Many of the ant-based studies will be discussed in the literature review (Chapter 2).

1.4 Solution Method

The aim of the thesis is to develop an examination timetabling solution method that can accommodate a mixture of hard and soft constraints. This solution method is segregated into two phases. The first phase exploits the ant feedback mechanism such that hard and soft constraints are tackled collectively. The second phase acts as a repair facility where further attempts are made to improve the condition of feasible timetables. During this subsequent phase, timetable feasibility is retained.

Initial experiments indicated that the solution method was very capable of producing timetables of quality that matched some benchmark methods. When the aim was to minimise the number of timeslots required, a basic (non-enhanced) ant-based method rivalled the known best on seven occasions (out of 12 data sets). The incorporation of algorithm based improvement strategies increased the success rate to 10 out of 12. The added inclusion of a hard constraint repair facility improves the rate further (11). When the aim was to allocate exams to a pre-specified number of timetables, the ability of the method was evident.

In phase two, only feasible solutions are eligible for improvement. Since it is determined that superior ant-based solutions require reduced repair time (thus validating the role of the ants), not all feasible timetables are improved (dependent on selection criteria used). Let us here refer to each ant-based solution eligible for improvement as a starting solution. When using some search process, it is shown that the neighbourhood definition around each starting solution does bear an influence on the ability to achieve superior final solutions. Also, it is demonstrated that the use of 19^{th} century evolutionary theory aids the search processes of the ants through encouraging wider exploration. Other means of broadening ant exploration are discussed.

1.5 Plan of Thesis

In Chapter 2, a review of related literature is presented. Chapter 3 examines the basic algorithm and gains insight into the appropriate parameter settings and construction heuristic that should be used to encourage better solution quality. Due to the inherent computational expense of the algorithm, methods of runtime conservation are then

discussed. This chapter lays the foundation for this thesis. Chapter 4 introduces first-order conflict on capacitated and uncapacitated levels. Conclusions are drawn regarding the success of the basic algorithm to generate feasible clash-free timetables. The benefit of methods of solution enhancement such as trail intensification, bias constants and hill climbing is discussed.

Chapter 5 presents Second-Order Conflict and discusses techniques that can be used to improve the Second-Order condition of a timetable. Firstly, components of the algorithm are modified to incorporate Second-Order characteristics. Secondly, additional factors, such as a bias constant and a weighted degree construction heuristic, are incorporated to improve quality. The investigation then concentrates on a two-trail philosophy, which allows information to pass between trails.

Chapter 6 demonstrates the additional benefit of an 'add-on' procedure. The Chapter considers three forms of evolution theory (Darwin, Baldwin and Lamarck) and assesses their influence on solution quality while utilising three add-on procedures (TSP, Local Search and Kempe Chains). The effectiveness of an ant based method as an examination scheduling tool is measured against Simulated Annealing.

Chapter 7 concludes the thesis and presents a modified version of the final algorithm to incorporate the well documented 'proximity costs' and results are compared against benchmark algorithms. Ideas for further research are also discussed.

Chapter 2

Literature Review

2.1 Introduction

Chapter 1 introduced examination timetabling and ant algorithms. A review, which discusses the literature belonging to these fields and closely related areas, is compiled in this chapter. The structure of this chapter is as follows.

Firstly, the work, as presented in Costa and Hertz (1997), that forms the basis of this thesis is detailed.

Secondly, the salient examination timetabling literature is studied. It would be impractical to cover all the work performed within this area due to volume of material available but the more relevant papers are included. This section is divided into five main subsections. The first subsection covers the common features of the examination timetabling problem. The second discusses a variety of single and multi phase methods that have been developed. The third subsection details the more recent exam based literature. The fourth addresses the use of timetable repair facilities such as local search and TSP. The final subsection surveys the literature published on the application of Genetic Algorithms to the examination timetabling problem, due to the similarities between them and ant systems. Both are population based, both are based on natural systems and both include means of learning from one generation to another, indicating that lessons can be learnt from literature relating to genetic algorithms.

Thirdly, a comprehensive review of ant-based literature is presented. This section attempts to step chronologically through the literature. It first regards the early considerations of AS and then observes extensions and developments of AS, which gave birth to Ant Colony Optimisation (ACO). To understand how ant algorithms are applied to real-life practical problems an account of ant applications is then presented.

2.2 ANTCOL

Costa and Hertz (1997) presented an implementation for graph colouring called ANTCOL. The aim of this algorithm is to use the learning powers of the ants to improve solution

quality of Assignment Type Problems with particular attention being paid to graph colouring. Since the examination timetabling problem has often been modelled as a graph colouring problem, it seems logical that ANTCOL could be applied as the basis of an examination timetabling solution method. The basic graph colouring method can be summarised as follows. Let G, with vertex set V and edge set E, be the graph describing the problem. The aim is to find a *q-colouring* with no adjacent vertices within the same colour group, with q as small as possible (hopefully equal to the chromatic number $\chi(G)$ – the minimum number of colours needed to colour the graph).

ANTCOL is described as follows. Ants are segregated into cycles and the aim of each ant is to generate a colouring while influenced by a mixture of pheromone, desirability and inherent randomisation that exists within the method. At some stage k, the experiences that have been accumulated during the previous k-1 stages are memorized in an n by n (where n refers to the number of vertices describing the problem) matrix M, which is updated at strategic times. For two non-adjacent vertices v_r and v_s, the value M_{rs} is the trail existing between v_r and v_s and represents the average quality of colourings obtained by allocating the same colour to v_r and v_s. For all adjacent vertices v_r and v_s, $M_{rs}=0$. To mimic the real-life parallel, trail levels are evaporated and this evaporation is represented by the parameter $(1-\rho)$ with $0 \leq \rho \leq 1$. It has been stated earlier that the use of evaporation in this analogous environment is to enhance the exploratory powers of the ants. Given m ants per cycle, the colourings generated within that time are $s_1,\ldots,s_m$ and $S_{rs} \subseteq \{s_1,\ldots,s_m\}$ denotes the subset of solutions in which v_r and v_s were allocated to the same colour group. Let q_t be the number of colours used in s_t $(1 \leq t \leq m)$. The update of M_{rs} in M is as follows in Equation 2.1.

$$M_{rs} = \rho . M_{rs} + \sum_{s_t \in S_{rs}} \frac{1}{q_t} \quad (2.1)$$

Equation 2.2 refers to the random proportional rule (RPR) and can be seen as a balance between a trail and desirability factor, which are raised to bias parameters α and β respectively. The role of these parameters is to stipulate the level of importance of the trail and desirability scores.

At a decision stage k, an ant chooses an uncoloured vertex i to be allocated colour j with probability $p(k,i,j)$ according to the roulette wheel philosophy, Goldberg (1989).

$$p(k,i,j) = \frac{\tau(s[k-1],i,j)^{\alpha}.\eta(s[k-1],i,j)^{\beta}}{\sum_{r \in J_i}\left[\tau(s[k-1],i,r)^{\alpha}.\eta(s[k-1],i,r)^{\beta}\right]} \quad (2.2)$$

where $s[k-1]$ represents a partial solution

J_i is the set of colours feasible for vertex i

α, β are the bias parameters for trail and visibility

The structure of the RPR does vary according to the style of construction that is used. These variants will be noted later.

$\tau(s[k-1],i,j)$ is the trail factor and represents the observed benefit of inserting vertex i into colour group j at stage k. It can be defined as follows.

$$\tau(s[k-1],i,j) = \begin{cases} \dfrac{\sum_{x \in V_j} M_{xi}}{|V_j|} & \text{if } V_j > 0 \\ 1 \text{ otherwise} & \end{cases} \quad (2.3)$$

where V_j refers to the number of vertices coloured j. If the colour j has some members then the trail factor is computed as the ratio of cumulative trails between a vertex i and members already assigned to colour j. If colour j is empty, then the trail factor takes a value of 1.

$\eta(s[k-1],i,j)$ represents the desirability score of vertex i at decision stage k. Costa and Hertz (1997) suggested eight suitable means of determining $\eta(s[k-1],i,j)$, which are based on two traditional graph colouring heuristics – the Recursive Largest Degree First (RLF), Leighton (1979), and Degree of Saturation (DSatur), Brélaz (1979). Both construction heuristic routines are detailed in Appendix 2.1. Simple descriptions of both heuristics are as follows. RLF builds the colours sequentially. A new colour is introduced

when no further insertions within the current colour group are possible. Vertices are allocated colours depending on their position within an ordered (according to degree) list and the non-violation of conflicts. DSatur builds the colours concurrently but only introduces a new colour when some vertex cannot be allocated to the current set of colours. Vertices are allocated their minimum feasible colour depending on their position within an ordered (according to the degree of saturation – see glossary of terms, Appendix 1.1) list. The lists associated with both heuristics are dynamically structured and are updated after the colouring of each vertex. The use of these heuristics in tandem with the ants will be referred to as ANT-RLF and ANT-DSATUR for ease of explanation.

2.2.1 ANTCOL graph colouring construction heuristics

ANT-RLF(Σ,Ω)

where Σ=1 and 2 and Ω=1, 2 and 3. The two procedures in Σ regard the selection of the first vertex within each colour group. The three procedures in Ω quantify the desirability (greedy philosophy) of inserting a vertex at some decision stage.

The six variants of ANT-RLF can be described in greater detail as follows.

Let W be the set of uncoloured vertices that can feasibly be coloured in the current colour group.
Let B be the set of uncoloured vertices that can no longer be coloured in the current group.
Let $deg_W(v)$ represent the degree of vertex v with respect to the vertices in set W.

The first vertex in each colour group is selected through:

$\Sigma = 1$ Random selection of $v \in W$

$\Sigma = 2$ $\max\{deg_W(v)\}$ with $v \in W$

Remaining vertices within each colour group are chosen by:

$\Omega = 1$ $deg_B(v)$ Degree of v with respect to infeasible uncoloured vertices.

$\Omega = 2$ $|W|$ - $deg_W(v)$ Difference between number of feasible uncoloured vertices and the degree of v with respect to the set of feasible uncoloured vertices.

$\Omega = 3$ $deg_{W \cup B}(v)$ Degree of v with respect to the uncoloured subgraph.

All components of Ω attempt to accommodate the larger degree vertices during the early stages of construction.

The probability of inserting a vertex i in colour j at decision stage k now becomes.

$$p(k,i,j) = \frac{\tau(s[k-1],i)^{\alpha} . \eta(s[k-1],i)^{\beta}}{\sum_{i \in W} [\tau(s[k-1],i)^{\alpha} . \eta(s[k-1],i)^{\beta}]} \quad (2.4)$$

Modification to RPR has arisen since we are no longer attempting to select an unknown colour j for an unknown vertex i. The anonymity of vertex i will remain but now there is prior knowledge of colour j due to the sequential nature of RLF.

ANT-DSATUR(ψ)

With $\psi=1$ and 2. When $\psi=1$, a vertex, when selected, is assigned to its minimum feasible colour c_{min}. At some decision stage, the trail score of an eligible vertex for colouring is computed by considering the trails with the coloured members of c_{min}. Meanwhile, when $\psi=2$, a vertex v does not necessarily receive the colour c_{min}, but rather some colour c (probabilistically biased according to trail levels) which satisfies $c_{min}(v) \leq c \leq q$, when q colours have already been used and $c_{min}(v)$ refers to the minimum feasible colour of vertex v.

The two variants of DSatur can be described as follows.

Let A be the set of uncoloured vertices.

Let $deg_A(v)$ be the degree of vertex v with respect to the set A.

The first vertex is selected by $\max\{deg_A(v)\}$. The expression $\eta(s[k-1],i)$ represents the degree of saturation for vertex i at decision stage k. The RPR for $\psi=1$ and $\psi=2$ are presented below:

When ψ=1

The probability of allocating c_{min} to vertex i at decision stage k is

$$p(k,i,c_{\min}) = \frac{\tau(s[k-1],i,c_{\min})^{\alpha}.\eta(s[k-1],i)^{\beta}}{\sum_{i\in A}\left[\tau(s[k-1],i,c_{\min})^{\alpha}.\eta(s[k-1],i)^{\beta}\right]} \quad (2.5)$$

When ψ=2

The probability of selecting vertex i at decision stage k is

$$p(k,i) = \frac{\eta(s[k-1],i)^{\beta}}{\sum_{i\in A}\eta(s[k-1],i)^{\beta}} \quad (2.6)$$

After the selection of vertex i, the probability of assigning it colour j

$$p(k,i,j) = \frac{\tau(s[k-1],i,j)^{\alpha}}{\sum_{j\in J_i}\tau(s[k-1],i,j)^{\alpha}} \quad (2.7)$$

where J_i is the set of feasible colours for vertex i. These eight construction heuristics will feature heavily in Chapter 3 and consequently, will be relabelled for ease of notation.

	Σ	Ω	Label
ANT-RLF	1	1	A
		2	B
		3	C
	2	1	D
		2	E
		3	F
ANT-DSATUR	ψ=1		G
	ψ=2		H

Table 2.1 – Labels used for construction heuristics for remainder of thesis.

2.3 Examination Timetabling

The basic examination timetabling problem can be described through the following formulation.

$$\min \sum_{i=1}^{E-1} \sum_{k=i+1}^{E} \sum_{j=1}^{P} c_{ik} x_{ij} x_{kj} \quad (2.8)$$

subject to

$$\sum_{i=1}^{E} A_i x_{ij} \leq B_j \quad \forall \text{ period } j \ (2.9)$$

$$\sum_{j=1}^{P} x_{ij} = 1 \quad \forall \text{ exam } i \quad (2.10)$$

$$x_{ij} \geq 0 \quad \forall \text{ exam } i, \text{ period } j \ (2.11)$$

where A_i = the number of students sitting exam i

B_j = the number of seats available in period j

c_{ik} = 1 if any students are taking both exams i and k, *0* otherwise

x_{ij} = 1 if exam i is scheduled for period j, *0* otherwise

E exams and P periods.

The objective function, Equation 2.8, minimises first-order conflict. The first constraint, Equation 2.9, ensures that the room constraint for period j is not violated. The second constraint, Equation 2.10, ensures that every exam is scheduled.

In practice, the examination timetabling problem is subject to a greater range of constraints. Typically, at the University of Wales Swansea the following criteria, obtained from Thompson (1995), must be adhered to in order to classify the timetable as useable.

Objectives

1. To minimise second-order conflict
2. To schedule large exams early

Constraints

3. There should be no first-order conflict
4. Time-window constraints, including pre-assignments are to be obeyed.
5. Ordering constraints are to be obeyed.
6. Certain pairs of exams are to be scheduled for the same time period.
7. Certain pairs of exams are not to be scheduled for the same time period.
8. Room constraints are to be obeyed.

2.3.1 Common Features

The following section details the aspects that are commonly discussed by authors with respect to the examination timetabling problem. Examples of how these concerns were addressed are given. General surveys on the examination timetabling problem can be found in Carter (1986) and Burke et al. (1995a).

Direct Conflict

An obvious first-order constraint is direct conflict. No student can physically sit two examinations at the same time. Therefore, any examination timetabling model must avoid such a possibility. It was Broder (1964) who first attempted to minimise the number of student conflicts, while, in a similar fashion, Cole (1964) presented an algorithm that minimised the number of time periods needed to ensure that no student had a clash. Foxley and Lockyer (1968) used the Cole paper as the basis of their investigation. The authors introduced a priority score for each exam that was determined by the number of clashes. The higher the priority score, the more important it is that the corresponding exam is allocated at an earlier stage in the timetable. The authors also introduced a second priority score for those exams that are known to be problematic to accommodate in the timetable. It was based on a scale of 1 to 15 with those exams with higher scores receiving overriding privileges with respect to timeslot allocation.

Mehta (1981) and Arani and Lofti (1989) consider fixed period timetables and attempt to produce a conflict-free schedule within this pre-determined length. The problem of minimising the number of periods necessary to produce clash free solutions has also been

tackled, for example Carter et al. (1996), Caramia et al. (2000), White and Xie (2000), Bullnheimer et al. (1997a).

There are occasions when all the exams cannot be timetabled within a pre-specified number of exam periods due to direct clash violations. Penalty functions are then introduced with the objective of minimising the number of conflicts. Dowsland (1990) and Mehta (1981) consider examples of this. Mehta (1981) introduces the 'compression of schedule' method. A time frame is collapsed and the exams/courses belonging to that time frame are distributed among other periods. The intention is to disperse the exams throughout the timetable to minimise direct clashes. The timeslot to collapse is chosen as follows. Let *w(i,j)* be the number of students taking both courses *i* and *j*. If course *i* is moved from some timeslot into a new timeslot Y_i then the number of conflicts is defined as

$$w(i,Y_i)=\sum_{j\in Y_i} w(i,j) \quad (2.11)$$

The timeslot X to be collapsed is the one that gives the minimum value of

$$\sum_{i\in X} Min\, w(i,Y_i) \quad Y_i \neq X \quad (2.12)$$

Exam *i* is moved to a timeslot Y_i such that $Min\ w(i,Y_i)$.

This method can be utilised when the predetermined number of exam periods is not sufficient. The timeslot with the lowest dispersal score is removed and the scheduler attempts to incorporate the affected exams within the remaining timeslots.

Bullnheimer (1997a) wrote that a modified QAP, namely the Quadratic Semi Assignment Problem (QSAP), could be used to model the examination timetabling problem. The difference between the QAP and the QSAP lies with the relaxation of a constraint that allows the allocation of more than one exam to any timeslot. The number of timeslots is not fixed but is subject to a penalty score to favour shorter timetables. In order to avoid first-order clashes, a very high penalty score is incorporated within a penalty function, which will be detailed in the following section.

Second Order Conflict

As stated above, the main objective for an examination scheduler is to eliminate first-order conflict. If this ambition is realised when utilising a fixed length exam period then some degree of freedom may exist to improve the *student acceptability* of the timetable. Many institutions aim to give students free periods between exams to aid their revision. This is most commonly managed by minimising the number of students who are required to sit two exams in a row, formally known as second-order conflict. White and Chan (1979) moved courses to alternate, feasible periods with fewer conflicting courses in the adjacent periods. Lofti and Cerveny (1991) and Desroches et al. (1978) rearranged the exam blocks through the TSP as a way to minimise second-order conflict. The TSP, used as a examination scheduling tool, will be detailed in Chapter 6.

The accommodation of higher-order conflicts has also been incorporated within examination scheduling solution methods. Carter et al. (1994) minimise the number of students that sat x or more exams in y successive time periods. Laporte and Desroches (1984) describe a penalty function of scheduling exams s time periods apart (described in Section 2.3.4). Mehta (1981) attempts to minimise the number of students that sit runs of three examinations in a row. Earlier pieces of work such as Foxley and Lockyer (1968) also tackled the student comfort problem. Each exam i is issued with a 'waiting periods' counter. After an exam i is inserted into the schedule, any other exam j that has at least one student in common with exam i is not allowed to be allocated within the next W periods.

The method (QSAP) proposed by Bullnheimer (1997a) is centred on the student's opinion of the timetable. The time between two periods, s and t, is defined as distance (d_{st}) and the underlying principle is to maximise the overall study time for students. A distance matrix $\tilde{d}_{st}$ is defined as in Equation 2.14.

$$\tilde{d}_{st} = \begin{cases} -M \text{ if } s = t \\ 0 \text{ if s and t are 'back - to - back'} \\ (d_{st})^{\alpha} \text{ otherwise} \end{cases} \quad (2.14)$$

$\tilde{d}_{st}$ draws a balance between spacing the exams for the students, the length of the timetable and minimising the occurrence of exams in consecutive timeslots. $\tilde{d}_{st}$ is aggregated according to the number of students affected and incorporated within the QSAP formulation with the objective of maximisation.

Large costs (-*M*) are incurred for first-order conflict violations. Zero costs are assigned to back-to-back incidences and higher-order conflicts are raised by $\alpha \in [0,1]$. The parameter α provides the scheduler with the ability to adjust to student comfort demands that are imposed by institutions (since the demands between institutions do vary). Setting $\alpha=0$ means that only back-to-back exams are regarded harmful to students, while $\alpha=1$ indicates that the impact of distance between exams per student is constant. Within these settings, α represents the *marginal product of study time*, which decays over time. Therefore, the costs presented in Equation 2.14 are scaled according to rate of acceptance by the students. Typically, allocating exams in the same timeslot that are common to students has a very large negative impact on the acceptance of a timetable, while larger spaces between exams are more welcomed and this is represented by the reward stipulated.

Bullnheimer proposes two variants of the QSAP. The first uses a fixed cost when institutions are indifferent regarding what timeslots should be free while the second utilises a cost that grows per extra timeslot and is used to shorten the length of the exam period. The authors claimed that the QSAP is suitable for small examination scheduling problems but will require some grouping algorithm, such as graph colouring, for larger problems that require the sitting of numerous exams during one sitting.

Balakrishnan (1991) used RLF to allocate feasible exams to timeslots and imposed tolerance limits to ensure that no more than a number x of students was were required sit back-to-back exams and 24-hour conflicts (number of exams that students sit during a 24-hour period).

Room Constraints

The number of students allocated to sit exams per period is often limited. Such a limit depends on the seating capacities of an institution. Such a constraint is of first-order concern and has to be observed. Arani and Lofti (1989) and White and Haddad (1983) ensure that the number of students assigned to take exams in any one period does not exceed the number of seats available. The allocation of exams to rooms was performed in a separate phase which simplifies the problem. If it is allowed for an exam to be split between rooms then a feasible solution is always obtainable.

Earlier work such as Wood (1968) acknowledged that seating students for exams was potentially problematic. Wood (1968) sorted exams according to size of room required. The exams were then segregated into groups and assigned to timeslots according to the largest degree first rule subject to feasibility and a selection of secondary objectives.

The increase in student numbers has made examination scheduling a more constrained problem due to the limited increase in seating facilities in response. The added pressures of modularisation and the desire to reduce the lengths of exam timetables have forced schedulers to solve the problem differently, Burke et al. (1995a). It is more common to allocate exams to rooms while generating the timetable and move away from the commonly used multi-phase approach, e.g. Carter et al. (1994). Balakrishnan (1991) discussed the more constrained problem at the Freeman School of Business, where only one exam per room is allowed. Consequently, it was imperative to monitor space utilisation during timetable construction. Lofti and Cerveny (1991) meanwhile, paid particular attention to the minimisation of the number of exams that require separate rooms. They formulated the problem of assigning exams to available rooms as a non-linear integer program. The authors ensured that the same exam, when split, was not sat on different campuses. Carter et al. (1994) guaranteed that split exams were placed in adjacent rooms.

Pre-Assignments and Time-Windows

Institutions often require certain exams to be pre-assigned or restricted to a subset of exam sessions due to a variety of reasons. Chan and White (1978) acknowledged the added

limitation. Burke et al. (1994b) stipulated that pre-assigned exams were to be allocated first. Balakrishnan (1991) demonstrated that the incorporation of period vertices, which form a clique, could be used to accommodate constraining exams. An edge between an exam and period vertex indicates conflict, therefore, preventing that exam from taking place during that period. Let there be t timeslots, the degree of a pre-assigned exam vertex with respect to the timeslot vertices will be $t-1$. Additionally, if an exam can be allocated to n timeslots then the degree of that exam vertex with respect to the period vertices is $t-n$.

Orderings

It is common for certain exams to have to follow others. Lofti and Cervany (1991) and Arani and Lofti (1989) state that at SUNYAB, certain pairs of exams have to be scheduled for successive periods. Burke et al. (1994b) caters for successive exams in the following way. Let exams A and B be ordered exams, with A preceding B. Exam A is a member of the graph, while B is maintained elsewhere. When A is added to the solution, B is immediately included and construction continues.

Simultaneous Exams

There are occasions when sets of exams have to be scheduled on the same day or even in the same room due to certain similarities between the exams. Mehta (1981) and Balakrishnan (1991) combine the information of relevant exams such that the exams are treated as one. On the contrary, institutional or departmental rules may dictate that certain exams cannot be sat during the same session. Mehta (1981) caters for this scenario through the creation of conflicts between relevant exams.

Large Exams Early

Many institutions prefer to schedule large enrolment courses earlier in the exam schedule to allow lecturers extra marking time. For example, Patnaik and Hosking (1985) schedule large exams early at the Asian Institute of Technology and Burke et al. (1995a) at Nottingham University. Johnson (1990) suggests that forcing early large enrolment exams early will conflict with the objective of minimising second-order conflict. A weighted linear function is used to deal with bigger exams at earlier stages of the construction

process. Consequently, these exams can be allocated to earlier timeslots and solutions are built around these large enrolment exams.

Number of Timeslots

Institutions do tend to have varying opinions about deciding on the number of timeslots that should be used. If an institution is flexible regarding the number of timeslots then the scheduler can construct a timetable that creates a balance between the number of slots and maximising study time for the students. However, as Bullnheimer (1997a) suggested, the benefit of extra study time for the student has a decaying effect. Burke et al. (1994a) and Bullnheimer (1997a) both suggest penalising each additional timeslot that is used above a pre-specified desired maximum, either through a fixed or an increasing cost.

Johnson (1990) orders the exams according to a linear function $Z_i=a*N_i + M_i$ with N_i representing the enrolment score of course i and M_i is the number of courses with which course i clashes. The weight a symbolises the relative importance of the two components. Courses are dealt with in descending order and allocated to feasible timeslots that increase the same-day (number of students that sit more than one exam in the same day) conflicts by the smallest measures. Solutions are improved, according to a same-day conflicts cost function, through an annealing (please see Appendix 6.7 for definition of Simulated Annealing) method. Timeslots are rearranged to minimise the number of students that sit back-to back exams.

2.3.2 Solution Methods

Examination timetabling has been tackled in a plethora of ways. In this section, we will divide the discussion into two main categories.

1. Single phase methods. The solution process is contained within one sweep of the algorithm.
2. Multi phase methods. The problem is divided into separate phases. Each phase has a different objective.

Single Phase Methods

Single phase heuristics are most commonly greedy single pass heuristics which tackle the problem in one iteration. These are not sophisticated techniques and have no ability to look ahead. Consequently, large, unwanted costs are typically incurred towards the end of the construction process. However, these heuristics are relatively easy to implement and were used in the early stages of automated examination timetabling. One of the earliest proposed methods was by Broder (1964), who minimises the number of direct conflicts using a simple greedy algorithm. Exams are ordered according to the student clashes and allocated to a period that is deemed the best. The determination of 'best' does vary according to the criteria used. In the case of a tie, allocation is performed on a random basis. The algorithm is run according to a number of different random streams in order to encourage solution variability.

The most important aspect of the method proposed by Cole (1964) involves the creation of a clash matrix C to inform the user whether two exams have at least one candidate in common. A value c_{ij} represents the clash score of exams i and j. If exams i and j have at least one student in common then $c_{ij}=1$ and $c_{ij}=0$ if there are no overlapping students between the two exams. Cole orders the exams in the same manner as Broder (1964) and refers to the clash matrix to deduce whether a particular allocation is viable. The work by Cole was furthered by Foxley and Lockyer (1968) who ordered the exams based on perceived levels of difficulty and importance. This led the way towards problem specific interaction to enhance solution quality. The user was also allowed to make manual adjustments.

Wood (1969) presented the similarity matrix where the *cell(i,j)* represents the similarity of vertices i and j. If vertices i and j are not connected then the similarity is equal to the number of neighbours common to i and j. The colouring procedure considers pairs of vertices at a time. The similarity matrix is scanned to find the greatest value. If vertices i and j have both been coloured then the next pair is considered. If both i and j have not been inserted then colouring depends on the degree of these vertices. If both degrees are less than the number of colour groups then the pair is ignored and another pair is considered. Otherwise, i and j are allocated to the colour group that allows feasible insertion. If not possible, a new colour group is started. If i has been coloured c_i and j is uncoloured then j

is added to c_i if the degree of j is less than the number of colour groups and feasible insertion is viable.

Graph colouring heuristics have played instrumental roles in many examination scheduling procedures and provide the backbone for the ANTCOL algorithm. Carter (1986) summarises many graph colouring heuristics. Earlier methods involved listing the vertices according to degree and filling the colour groups from the top of the list. Examples are Welsh and Powell (1967) and Peck and Williams (1966). The former builds the colours concurrently and allocates vertices to the lowest feasible colour while the latter fills the colour groups sequentially and attempts to fill the timeslots as much as possible without any direct conflicts. After the completion of each colour group, allocation to the next colour group starts at the top of the list. These approaches were overshadowed by similar, but dynamic methods that updated the lists according to the uncoloured subgraph and some allowed backtracking facilities to improve colourings. Dunstan (1976) presented the *Largest degree first recursive: fill from top* heuristic. After the colouring of a vertex, it is removed from the graph. The degree of each uncoloured vertex is recalculated with respect to the uncoloured subgraph and the degree list is re-sorted. Matula et al. (1972) introduced *Smallest degree last recursive with interchange* which lists the vertices in descending order according to degree. Vertices at the bottom of the list are removed from the graph and the degree of each eligible vertex is recalculated and the list resorted. When all vertices have been coloured, the original removed vertices are then coloured. If after the construction of c colours, there exists an uncoloured vertex v_i that clashes with at least one member v_j of each colour group then a vertex swap is needed, which is defined as follows. If there exists a colour c that contains one neighbour, v_j, of v_i then recolour v_j if feasible and insert v_i into c. Otherwise, the authors suggested a *bichromatic* interchange. For each colour group c, determine the set of neighbours V_c of v_j. If the set V_c does not conflict with vertex v_i or any other vertices in c, then interchange set V_c with v_i. Leighton (1979) and Brélaz (1979) detail important graph colouring heuristics. These are presented in Appendix 2.1 and mentioned in Section 2.2 where the ANTCOL algorithm is detailed.

A similar type of enhancement strategy was detailed in Carter (1986) and Carter et al. (1996). Backtracking was used to amend for 'earlier errors'. This method attempts to accommodate 'difficult' exams through the removal of other exams from the schedule. These displaced exams are then re-inserted into feasible timeslots if possible. Results

presented in Carter et al. (1996) indicate that backtracking is worthwhile when applied to the solutions returned through single pass heuristics. The average reduction in schedule length is 50%. Before backtracking is implemented, exams are inserted in the timetable based on some exam list pre-ordering. Five sorting criteria were compared. Some traditional techniques were used such as largest degree and saturation degree. However, the authors also used largest enrolment, random ordering and largest weighted degree (largest degree weighted by student enrolments). It was stated that ordering by saturation degree led to the best results. The results do indicate the importance of good pre-ordering of exams. The authors also include an initial phase, which colours the vertices belonging to the maximum clique. It was recognised that this procedure was beneficial to solution quality given that examination timetabling problems generally have large maximal cliques, thus increasing colouring difficulties. Carter at al. also introduced a cost function that is based on the penalty function detailed by Laporte and Desroches (1984). The Carter et al. cost function will be used in Chapter 7.

Kiaer and Yellen (1992) presented a single-pass heuristic for the university course timetabling problem using weighted graph colouring. The vertices represent the courses, while a connecting edge between two vertices indicates conflict. Whereas standard graph colouring would demand a colouring that allocates no connected vertices to the same colour group, weighted graph colouring relaxes this limitation. Each edge is weighted according to the criteria used e.g. number of students in common and the objective is to obtain a colouring that minimises the overall weighting. The algorithm colours each vertex in turn to the period of lowest cost, with appropriately high costs being used to ensure that constraints are not broken. See Dowsland (1990) for a similar study.

Čangalović and Schreuder (1991) proposed a weighted graph colouring model for the timetabling problem, with vertices assigned weights based on the length of the lecture. Two vertices are connected by an edge if the corresponding classes have teachers or rooms in common. Assigning a colour to a vertex represents the hour that a class will take place and the vertex weight indicates how many colours (hours) need to be assigned to that vertex. The objective is to find an interval *k-colouring* that does not violate any conflicts.

Graph colouring heuristics exist due to the problem at hand being NP-Hard, however there are exact procedures. The use of an exact method is inappropriate for examination

timetabling due to typical problem size. Literature detailing exact methods can be found in Ellis and Lepolesa (1989) and Brown (1972). Modification to Brown (1972) can be found in Brélaz (1979) and corrections to this can be seen in Kubale and Kusz (1983) and Peemöller (1983).

Multiple Phased Approaches

Multiple phase methods break the problem down into several parts and solve each individually. The solution(s) achieved after each stage is/are retained and the requirements of the next stage are incorporated and another (set of) solution(s) is/are attained.

Arani and Lotfi (1989) detailed a three-phase approach for the examination timetabling problem. Phase 1 assigns exams to blocks so as to minimise direct conflicts. The number of blocks is equal to the number of timeslots. This problem is modelled as a QAP without column constraints and modelled heuristically in the following way. The vertices are ordered according to the weighted degree (product of degree of vertex and the number of students taking the exam). Each exam is assigned to the first available session or the timeslot that creates minimum conflict, which is the objective in the first phase. The second phase designates which exam blocks are to be placed within the same exam day. The authors outlined suitable approaches based upon the number of exam slots within each day and the attitudes of the institutions. If the last session of one day can be considered adjacent to the first session of the following day then the task of minimising second-order conflict can be modelled as a TSP. However, this was not the case at the University of New York at Buffalo (SUNYAB) and the authors used a different approach. The problem was modelled as a set-covering problem and was solved using a Lagrangian Relaxation based method. The objective was to minimise the number of students that sit two or three consecutive exams in each day. The third and final phase organises the exam blocks within each exam day to minimise consecutive exams for students. If there are three exam periods within each exam day then the two exam blocks that creates the most conflict can be assigned to the first and last blocks of each day. Therefore, such an amendment is quite trivial. However, if the number of exam periods within each day exceeds three, the problem can be modelled as a TSP.

Lotfi and Cervany (1991) extend the work discussed by Arani and Lotfi (1989) by incorporating some additional constraints and introducing a fourth phase, which assigns

exams to rooms. Phase 1 is still modelled as a QAP but, in addition, the number of exams per block is limited to 100 and no lecturer can be in charge of two exams within the same block. Phase 2 assigns exam blocks to exam days. This task was modelled as a QAP without column constraints to produce an initial solution and swaps were used to enhance quality. Phase 3 organises the exam blocks within exam days and is modelled, as in Arani and Lotfi (1989), as a TSP. It is solved via the 'next nearest city' heuristic as described in Papadimitriou and Steiglitz (1982). The 'new' fourth phase assigns exams to rooms to maximise the space utilisation. A greedy heuristic is utilised, which minimises the number of split exams while observing university regulations at SUNYAB, which requires that a final exam must be held in a room with a seating capacity of at least twice the number of candidates.

Balakrishnan et al. (1992) minimises the number of students that are required to write consecutive exams on the same day through a network flow model. A node *(i,t)* symbolises the assignment of exam group *i* to timeslot *t*. The edges define the various predecessors and successors and the associated weight represents the impact on the second-order quality of the timetable. For example, an edge that connects node *(i, t)* and node *(j, t+1)* indicates that exam group *j* immediately follows exam group *i*. The number of students in common with the two exam groups weights this edge. If two exams are scheduled for the same day then the weight is δ but zero if crossing two days. A shortest path matrix *L* is constructed where *L(i,t)* represents the length of the shortest path from the source to the sink, which includes node *(i,t)*. *L* is calculated using dynamic programming recursions. The value *L(i,t)* can be taken as a lower bound on the minimum number of back-to-back conflicts obtained if exam group *i* is allocated to timeslot *t*. The minimum value in any column of nodes is the lower bound and a tighter bound can be obtained through taking the maximum minimum over the columns. Any value *L(i,t)* that is lower than this tighter lower bound is increased to that value. The assignment problem can be used to assign exam groups to timeslots to minimise the sum of the selected paths, which acts as an upper bound. Additionally, a new lower bound can be derived from the average path length from the assignment problem. Lagrangian relaxation is then used to improve the lower bound. The relaxed problem is guaranteed to obtain a feasible solution and any solution value is a lower bound to the original problem. A subgradient technique is used to improve the Lagrangian multipliers and consequently, new *L(i,t)* values. This procedure is recursive and terminates when a

satisfactory solution has been generated. The authors claim that this approach outperforms the TSP.

When institutions, such as SUNYAB, hold three exam sessions per day, students may experience the possibility of sitting two exams in the same day. Arani et al. (1988) attempt to minimise this occurrence through modelling it as a set-covering problem with additional side constraints. The objective function represents the sum of the students that are to sit two or more exams during the same day and is minimised subject to two constraints. The first constraint stipulates that each exam block has to be assigned to a timeslot and the second constraint ensures that the number of exam days equals the predetermined number of exam days. The form, in which the first constraint is applied, if at all, determines the nature of the solution method i.e. set covering or set-partitioning problem. The authors recognised that the relaxation of the first constraint through Lagrangian multipliers (that forces each block to be assigned to exactly one timeslot) tends to produce better bounds at each node in the branch and bound tree. Bounds can be produced efficiently by inspection. Two side constraints are imposed to improve the quality of the bound. The first finds the two cheapest arrangements. The more expensive arrangement is removed if it contains an exam block included in the cheapest arrangement. The cheaper arrangement is then added to the solution and the procedure continues. An arrangement may remove another arrangement from inclusion within the solution on only one occasion. Otherwise, the lower bound becomes invalid. The second side constraint attempts to exclude high cost arrangements that have at least one exam block in common with those arrangements that are already fixed in the solution. The initial multipliers are computed according to the dual variables obtained through the LP relaxation of the objective function. The multipliers are updated according to specified step sizes and over a maximum number of iterations. This relaxation method is shown to improve performance.

Other related studies include Tillet (1975) who assigned teachers to courses using integer programming and Carter (1989) who tackles the classroom assignment problem using Lagrangian Relaxation. Tripathy (1980) also used Lagrangian relaxation within a branch-and-bound procedure for modest size problems based on real data. Results were compared against the heuristic presented in Barham and Westwood (1978).

2.3.3 Recent Developments

Over the last decade, timetabling algorithms have become even more sophisticated and wide ranging. This section will discuss a subset of recent literature. Thompson and Dowsland (1998) tackled the examination timetabling problem using a two phased approach. The first phase obtains a feasible solution and the second seeks to find an improvement with respect to secondary objectives and soft constraints while maintaining feasibility. The authors use simulated annealing to explore the solution space and define various neighbourhoods. It is demonstrated that the use of the graph-theoretic concept of Kempe Chains extends the neighbourhood in phase two, Morgenstern (1989a), and leads to superior solution quality. Thompson and Dowsland (1998) stressed that the neighbourhood definition for phase two was very important for overall solution quality.

Morgenstern (1989a) suggested the Kempe Chain neighbourhood for the graph colouring problem. The great advantage of this type of neighbourhood is the maintenance of feasibility, which will form a pivotal role in Chapter 6. Morgenstern used a sampling mechanism that is biased towards the selection of vertices from small colour classes. However, a Kempe chain move is as likely to increase the size of a chosen class as decrease it, and therefore, does not necessarily aid in the removal of small colour classes.

Thompson and Dowsland (1998) used problem specific decisions for phase one of their solution method as presented in Chams et al. (1987). The solution space consists of all partitions of vertices into k colour classes. The neighbourhood is the set of solutions produced by changing the colour of one vertex. The cost function is equal to the number of edges between vertices in the same colour class and the starting solution is produced randomly.

Merlot et al. (2002) use simulated annealing with Kempe Chains within their hybrid algorithm for the examination timetabling problem. The authors presented a three-phase strategy. The first phase uses constraint programming to obtain a feasible timetable. Allocation of exams to timeslots is performed according to some criteria i.e. degree, enrolled students. If the feasible allocation of some exams is impossible within a stipulated set of timeslots then backtracking is used to form gaps (removal of exams) within the timetable to allow entry of these unallocated exams and subsequent moves will then

attempt to re-accommodate these removed exams. The second phase improves solution quality with respect to secondary objectives. Simulated Annealing is used to search the solution space and Kempe Chains were incorporated to extend the neighbourhood. A third phase uses a hill climbing strategy to ensure that the final solution is a local optimum. The results presented are competitive with the benchmark solutions of Carter et al. (1996) and Caramia et al. (2000).

Caramia et al. (2000) presents a set of algorithms for the examination timetabling problem. The *greedy scheduler* allocates exams, ordered according to degree, to the lowest feasible timeslot, while the *penalty decreaser* spaces out the exams evenly while keeping the number of timeslots fixed. If the *penalty decreaser* is not successful then the *penalty trader* is used. The *penalty trader* trades off penalties for timeslots. This facility decides which exams (ordered according to benefit) should be moved to additional timeslots. Additional aspects such as the *checkpoint scheme* and *bridging priorities* were included to allow for wider exploration. The former stops the algorithm at certain steps, releases memory of previous searches and starts a new local search. The latter prevents searching similar areas of the solution space.

Burke and Petrovic (2002) indicate that the future of automated timetabling lies with the use of hyper-heuristic methods. The objective of a hyper-heuristic is to design an algorithm that will select the most appropriate heuristic to carry out a certain task based on the environment i.e. conditions of the search, problem to be solved. Ross et al. (1998) suggest that genetic algorithms could be used to select appropriate timetabling heuristics.

2.3.4 Local search and TSP

After a solution method has been used to construct a timetable, improvement techniques are used to enhance solution quality. Local search and TSP methods are often used in this area and examples of the application of these strategies are detailed here.

Local search is a neighbourhood search technique, which begins with some initial solution and continually improves until no further improvement is possible. Two main implementations of local search exist. Firstly, random descent chooses a neighbouring solution at random and assesses its quality with respect to some predefined cost function. If

the cost of the newly generated solution is more attractive it becomes the current solution. If not, another neighbour is sampled. The search terminates if no improvement has been detected for a number of sampled neighbours. With Steepest Descent, the entire neighbourhood is evaluated and the best move is chosen. The procedure stops when no improvement is possible. Local search is quick and easy to implement as a search technique. However, the search gets trapped in the first local minimum, thus often returning poor solution quality.

The TSP is a traditional combinatorial optimisation problem that has often been used as a testbed for ant algorithms. The TSP will feature prominently in Chapter 6 and has often been used, as stated above, as an approach to improve the student comfort dimension of an examination timetable. Examples of application of the TSP in the examination timetabling problem are as follows. White and Chan (1979) solved the examination timetabling problem as follows. Initially, clash-free exam groups were constructed using a relatively simplistic heuristic. These exam groups were allocated to timeslots using the TSP model to minimise second-order conflict. Steepest Descent was then used to determine whether any exams could be inserted into alternative timeslots. A move would be made if second-order improvement is detected and no direct conflict violation would result. The authors implemented a further phase of improvement. The swapping of each pair of exams was evaluated in order to see whether any swap would improve the second-order condition of the timetable, while not violating direct conflict. On each occasion, the best swap was selected and the solution updated accordingly.

Colijn (1997) considered the impact of higher-order conflicts. The author felt that students who were required to sit three or four consecutive exams suffered considerably greater emotional stress than those sitting back-to-back exams. The higher-order problem was still modelled as a TSP but storage facilities become problematic. Typically, $C^3_{i,j,k}$ represents the number of students writing exams in period i, j and k. To store all permutations of i, j and k would require a three dimensional matrix of size t^3 (let t represent the number of timeslots). Meanwhile, White and Haddad (1983) use White and Chan (1979) as the basis for their work on the day and evening courses problem. The two sets of courses are scheduled independently and merged once constructed. A random descent procedure leads to a reduction in the number of students that sit two exams in a day.

Other examples of utilising local search and/or TSP to improve solution quality are as follows. Patnaik and Hosking (1985) presented a solution method to minimise the number of clashes in an examination timetable using APL (**A** Programming **L**anguage). The problem was modelled using graph colouring and a greedy heuristic that allocates an exam to the feasible colour that is least likely to be required by a neighbour. Attempts were made to minimise the number of direct conflicts subject to the number of timeslots being fixed. Local search was then used to improve solution quality. Eiselt and Laporte (1987) use random descent to improve the solutions that were generated by their algorithm. Work presented in this paper was based on Laporte and Desroches (1984). The most notable aspect of Laporte and Desroches (1984) resides with the use of aversion and proximity costs. The former is explained as follows. For each examination e and each timeslot t, the scheduler can quantify an aversion cost p_{et}. The larger the aversion cost, the less attractive the insertion of exam e within timeslot t. Proximity costs are used to space the exams evenly through the timetable in order to reduce second and higher conflicts for the students. The program (known as HORHEC) assigns each pair of examinations with a cost w_s, where s represents the number of timeslots separating the two exams. The program deals with five costs, defined as follows.

$$w_1=16,\ w_2=8,\ w_3=4,\ w_4=2,\ w_5=1$$

If a student is required to sit back-to-back exams then a penalty of 16 is incurred, two-timeslot difference presents a cost of 8 and so on. The overall cost function is a weighted sum of the aversion and proximity costs.

The proximity costs can also be found in later literature e.g. Carter et al. (1996), Caramia et al. (2000), and have been used as a standard cost system to compare against standard algorithms. This cost function will be used in conjunction with work in Chapter 7.

Burke et al. (1995c) presented a memetic algorithm that utilised local search to improve the solutions constructed via genetic algorithms. Additionally, Section 2.4 shows that local search is beneficial to the success of ant algorithms. Such evidence indicates that applying AS to the examination timetabling will also yield superior solution quality through local search or some other repair facility (see Chapter 6).

The use of local search may not achieve the desired solution quality since searches get trapped in local minima. Several variants have been proposed to overcome this disadvantage. Multi start algorithms investigate different parts of the solution space but do not benefit from any gathered knowledge. More sophisticated methods such as Simulated Annealing, Eglese (1990), and Tabu Search, Glover (1989), have been investigated and gained much popularity over the last couple of decades.

2.3.5 Genetic algorithms

Recent years have seen the development of evolutionary methods as solution methods. Focus is naturally placed on Ant Algorithms throughout this thesis, but, here, attention is paid to another population based meta-heuristic, Genetic Algorithms. A formal definition of this approach can be found in Appendix 2.2 and a detailed discussion coupled with application of Genetic Algorithms can be found in Dowsland (1996).

A standard application of GA to the examination timetabling problem would be as in Burke et al. (1994a). Initially, a random population of feasible timetables are created using a variation on a graph colouring heuristic. Each timetable is then assessed according to the fitness criterion used e.g. the length of the timetable, linear penalty function. Timetables are then chosen (according to rules such as roulette, tournament, top percent etc...) to be the basis of the next generation, with good timetables having a greater probability of being selected. Burke et al. (1994a) use an advanced crossover function which selected two random crossover points i and j. Exam assignments x_1, ..., x_{i-1} and x_j, ...,x_n are taken from *parent 1*. The remaining exams x_i, ...,x_{j-1} are allocated as close to their assignments in *parent 2* as possible. A mutation operator is applied to randomly change the period (and the room), while maintaining feasibility.

There are essentially two types of representation that have been used, which vary in the way that the timetable is encoded. Corne et al. (1993) tackle a problem at Edinburgh University involving around 40 exams to be scheduled into 28 time periods. The authors used a traditional approach (direct representation) where each gene represents the time at which each particular exam takes place. Uniform crossover and standard mutation produce good results using 300 generations and a population size of 50. However, this examination scheduling problem is relatively small. Paechter et al. (1994) use a *place-and-seek* method

to allow the gene to not only specify when the exam is taken but search for a new period if the exam is causing conflict after crossover (implicit representation). If the exam cannot be placed, it is left unscheduled and thus, prevents infeasible timetables being part of the population. Burke et al. (1994a) argue that when infeasible timetables are allowed there is a danger of conflicting information and poor solution quality may result, however Burke et al. (1995d) point out that restricted search spaces may make the best timetable harder to obtain.

Burke et al. (1994a) did not fix the number of timeslots but did penalise each additional timeslot above a *preferred* maximum within a linear penalty function (also accommodated the number of back-to-backs and spare seating capacity per timeslot). If a correct balance of weights is used then the search would bias towards shorter timetables. Burke et al. (1995b) discussed a modified crossover operator that suits the constrained environment of the examination timetabling problem. The operator takes each period in turn and allocates exams to each period (for the child) that featured in the same period for both parents. Other exams are then allocated to periods (dependent on feasibility) according to some graph colouring heuristic. Unplaced exams are then considered for the next period.

Burke et al. (1995d) presented specially designed operators to suit the examination timetabling problem - *Late* (to reduce length of timetable), S*pread* (to aid the second-order condition of the timetable) and *Proximity* (to maintain the proximity of the exams if it is recognised that the distance between exams is beneficial). Combinations of these operators have also been proposed. However, these operators do not offer any significant additional constructive power, Burke and Petrovic (2002).

Ross and Corne (1995) concluded that Simulated Annealing and Stochastic Hillclimbing outperformed the basic Genetic Algorithm model. It has been recognised that the use of local search is needed to repair chromosomes before the information is injected into the population. See Jog et al. (1989) and Radcliffe and Surry (1994) for discussions. Moscato and Norman (1992) first coined the term memetic algorithm to describe a hybridisation of an evolutionary algorithm and local search. The motivation derived from the notion of a meme as a unit of information that reproduces itself as people exchange ideas. The main difference between a meme and a gene is as follows. Before a meme is passed on to offspring it is adapted according to problem knowledge (Lamarckian evolution theory, see

Chapter 6). Conversely, genes get passed on whole. Reeves (1994) suggest that the decision to apply local search to a solution should be a stochastic one due to the characteristic difference that exists between the parent and offspring solutions. Burke et al. (1995c) uses a combination of mutation and local search to search the solution space. The authors detail two forms of mutation operators, light and heavy. The former moves a few individual exams into alternative feasible timeslots, while the latter focuses on timeslots where large direct conflicts exist.

To fully evaluate the quality of a timetable that attempts to accommodate all the constraints, a multi-weighted fitness function is sometimes used, Corne et al. (1994), Colorni et al. (1990). A typical example of a fitness function is presented in Burke et al. (1998) and Burke and Petrovic (2002).

$$Cost(t)=5000*unscheduled(t)+3*sameDay(t)+overnight(t) \quad (2.13)$$

where *unscheduled(t)* stores the number of exams not scheduled in a valid period, *sameDay(t)* and *overnight(t)* holds the number of second-order conflicts during the same day or overnight respectively. In this instance, the authors stipulate that a student who encounters back-to-back exams during the same day is three times worse than successive exams which include an overnight break. Ross et al. (1994) also acknowledges that a weighted linear function is appropriate, however the selection of weights is crucial. Typically, assigning weights of 1 and 0.1 for first and second-order clashes respectively would suggest that the removal of eleven second-order clashes is marginally better than the removal of one first-order clash. The authors suggested that weights of 1 and 0.01 would be more reflective of the balance between the two priorities. However, Di Gaspero and Schaerf (2000) state that fixed weights do not work well when applying Tabu Search to the examination timetabling problem. To counteract this, the authors suggested using a *shifting penalty* mechanism, which determines the appropriate times during a search when a weight w should be scaled by some factor γ.

2.4 Ant Algorithms

The first examples of Ant Algorithms can be associated with the Ant-Density, Ant-Quantity and Ant-Cycle algorithms (all discussed in Chapter 1). Dorigo et al. (1991) is the forerunning paper on these procedures and proved that Ant-Cycle is the superior technique and consequently, acts as the basis for subsequent research in the field of Ant Algorithms. This section of the literature review charts the various Ant Algorithms and applications that have been researched since its introduction in 1991. The structure of this section is as follows. Initially, we regard the issues that concerned the researchers during the early stages of development e.g. parameters, synergistic role. Such concerns play pivotal roles now and need to be addressed here. Then we consider extensions to the algorithm that have been incorporated to improve the search abilities of the algorithm(s) such as elitism and ranking ants. To illustrate the use of ants in practice, applications of Ant Algorithms to combinatorial optimisation problems are reviewed. Many of the studies discussed in the following section refer to the TSP, which is a traditional combinatorial optimisation problem that has been extensively studied in literature. This, coupled with the natural parallelism between the routing problems and the basic analogy of ants suggests that the TSP is a suitable testbed for Ant Algorithms. All examples will use the TSP unless stated otherwise.

2.4.1 Early considerations

This subsection addresses the main areas of interest during the infancy of Ant Algorithms.

Parameter Settings

The success of the search depends upon the choice of parameter settings. The standard parameters α, β and $(1-\rho)$ refer to the level of trail bias, desirability bias and evaporation respectively. These parameters are normally used as constants during a search and can be selected by the user, however White et al. (1997) do suggest that the parameters could be altered during a run through the use of Genetic Algorithms. With respect to the bias parameters, it is important to obtain a suitable balance. For example, high levels of α places too much emphasis on the trail and will potentially limit ant exploration and the ants

will exhibit stagnation-like behaviour (all ants construct the same path). This is undesirable since no improved solution will be found after the onset of stagnation. Conversely, using too low a value of α reduces the algorithm to a stochastic multigreedy algorithm and the impact of the ants' feedback will be lost and no exploitation will take place. The inclusion of the desirability function exploits the inherent structure of a problem type. With respect to the TSP, the knowledge of distances between cities can be advantageous in a greedy manner and its inclusion is worthwhile. However, stipulating a high desirability bias parameter will also reduce the relative impact of trail.

Dorigo et al. (1991) investigated the impact of parameter settings on solution quality. Tests were performed for Ant-Density, Ant-Quantity and Ant-Cycle algorithms for a range of α, β and ρ. It is observed that there is a significant disparity in solution quality for $\alpha,\beta=0$ and $\alpha,\beta>0$ for all algorithm types. The preferred settings are $\alpha=1$ and $\beta=10$ for Ant-Density, $\alpha=0.5,\beta=20$ for Ant-Quantity and $\alpha=1,\beta=5$ for Ant Cycle. With respect to pheromone evaporation, the preferred settings of ρ are *0.999*, *0.999* and *0.5* for Ant-Density, Ant-Quantity and Ant-Cycle respectively. The authors stated the different preference of ρ could be attributed to the style of reward these algorithms impose. Ant-Density and Ant-Quantity replenish trails on a local level, which is a function of the desirability function, thereby emphasizing the greedy aspect of the algorithms. In contrast, the reward function of the Ant-Cycle technique represents the tour length of a solution (global). After a number of updates, the ants exploit this global information to the benefit of solution quality. However, the search process also benefits from some exploration. Thus, to prevent the trail becoming too dominant and to allow the use of desirability information, a reduced measure of ρ seems fairly intuitive. Dorigo et al. (1991) added that high values of α encourages the algorithm to perform stagnantly from relatively early stages of the search and limits the ability to construct competitive solutions. In addition, they observed that if not enough importance was allocated to the trail then the algorithm could not obtain very good solutions either. In practice, there is not a definite rule for these parameters and are normally problem specific. The α and β settings chosen by Costa and Hertz (1997) for the ANTCOL algorithm are discussed in Chapter 3.

Synergistic Effect

Dorigo et al. (1991) investigated the influence of population size on solution quality. It was observed that too few ants were not appropriate and optimality was reached when the number of ants per cycle approached the number of cities in the graph. Beyond this optimality point, solution quality improvements dampen and consequently, runtimes unnecessarily inflate. The authors also suggested that ant-starting points should not be limited to one city, but rather be uniformly distributed. So, if the number of ants equals the number of cities then one ant should start from each city.

2.4.2 Enhancements of the Ant System

In this section, modifications that have been made to AS to enhance solution quality are discussed.

Elitism

De Jong (1975) first applied the concept of elitism in his PhD thesis, which has proved pivotal in the development in GA theory. The basic idea behind elitism is to give extra emphasis to the best solutions found. Dorigo et al. (1991) used the concept of elitism and introduced elitist ants (AS_{elite}), which is equivalent to sending a certain number of ants over the best solution. The update philosophy is as follows.

$$\tau_{ij}(t+1) = \rho.\tau_{ij}(t) + \Delta\tau_{ij} + \Delta\tau_{ij}^{*} \quad (2.15)$$

$$\text{where } \Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \text{ and } \Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{L^k} \text{ if ant } k \text{ travels on edge } (i,j) \\ 0 \text{ otherwise} \end{cases} \quad (2.16)$$

$$\text{and } \Delta\tau_{ij}^{*} = \begin{cases} \sigma.\frac{Q}{L^*} \text{ if edge}(i,j) \text{is part of the best solution found} \\ 0 \text{ otherwise} \end{cases} \quad (2.17)$$

where $\Delta\tau_{ij}^{*}$ is the increase of trail level on edge (i,j) caused by the elitist ants
σ is the number of elitist ants
L^{*} is the tour length of best solution found
Q is a fixed constant

White et al. (2003) proposed that the number of elitist ants used should be equal to $(1/6)^{th}$ of the number of cities (vertices in problem). However, the authors noted that experiments were only performed on smaller data sets and further testing would be required to test the $(1/6)^{th}$ rule on larger data sets. The authors suggested that altering elitism from a global perspective to a local one encourages superior solution quality. Such an approach requires extra time for solution convergence but the search has a great probability of obtaining the global optimum due to a reduced chance of getting trapped in local minima during the early stages of the search. Bullnheimer et al. (1999) suggested that the number of elitist ants should equal the number of cities. Dorigo et al. (1991) indicated that a range of elitist ants could be used. Tests performed on a 30 city problem showed that setting the number of elitist ants between 2 and 16 returned the optimal solution, while using no elitist ants returned relatively poor solutions. Therefore, the opinions of the authors do vary and researchers are advised to perform their own fine-tuning.

Bullnheimer et al. (1999) noted a valid drawback of the elitist system. If solution quality increases and the differences between individuals decrease, then consequently the difference in selection probabilities also decreases and, in return, exploitation is not as potent as desired. If this occurs then the ants tend to discover good but sub-optimal solutions. The groundbreaking work presented by De Jong (1975) demonstrated that elitism is beneficial to overall solution quality. Elitism will play a vital role in this thesis.

Ranking Ants

Bullnheimer at al. (1999) claimed that elitism directs the search process towards suboptimality and consequently introduced the concept of ranking ants. AS_{rank} allows further differentiation between solutions, which the authors claimed reduces the danger of inflated trails on wrong paths because they are part of sub-optimal solutions. The framework of AS_{rank} is as follows. After m ants construct solutions, the ants are ranked according to fitness. The contribution of an ant to the trail level is weighted according to its' rank μ, while only the best ω ants contribute to the trail. The best ant is equipped with a weight σ and the μ^{th} ant uses a weight $(\sigma-\mu)$ with $\omega = \mu-1$, therefore the minimum weight is set as one. The update philosophy is as follows.

$$\tau_{ij}(t+1) = \rho.\tau_{ij} + \Delta\tau_{ij} + \Delta\tau_{ij}^{*} \quad (2.18)$$

where $\Delta\tau_{ij} = \sum_{k=1}^{\sigma-1} \Delta\tau_{ij}^{\ k}$ and

$$\Delta\tau_{ij}^{\ k} = \begin{cases} (\sigma - \mu)\frac{Q}{L^u} \text{ if the } u^{th} \text{ best ant travels on edge } (i,j) \\ 0 \text{ otherwise} \end{cases} \quad (2.19)$$

$$\text{and } \Delta\tau_{ij}^{\ *} = \begin{cases} \sigma.\frac{Q}{L^*} \text{ if edge}(i,j) \text{ is part of the best solution found} \\ 0 \text{ otherwise} \end{cases} \quad (2.20)$$

where $\Delta\tau_{ij}^{*}$ is the increase of trail level on edge (i,j) caused by the elitist ants
σ is the number of elitist ants
L^* is the tour length of best solution found
μ is the ranking index

Comparative tests for basic AS, AS_{elite}, AS_{rank}, SA and GA methods were presented in Bullnheimer et al. (1999). For AS_{elite} the number of elitist ants was set to the number of cities, while six elitist ants (for problem sizes varying from 30 cities to 132 cities) were used for AS_{rank}. Five test problems were tackled and AS_{elite} and AS_{rank} outperformed SA and GA methods on the larger data sets and were consistently better than basic AS, with AS_{rank} performing marginally better than AS_{elite}.

Parallelisation Strategies

Bullnheimer et al. (1997d) extended the inherent parallelism of the AS by introducing a *worker* system. A worker can be defined as a processing unit and this approach employs numerous workers. Ants are sent out from each worker, solutions are constructed and the associated fitness evaluated and passed to the master unit, which assesses the information and the elite information is passed back to the workers and used in subsequent investigations. The main objective behind parallelisation is to achieve good quality solutions in reduced computational times. Bullnheimer et al. discussed synchronous and partially asynchronous strategies. A synchronised system dictates that workers pass information to the master unit after each iteration. The master unit evaluates the solutions and selects the best, then updates the trails appropriately and passes the information to the workers. Once accomplished, ants perform more explorations. An asynchronised system

reduces the number of worker-master slave information exchanges that are performed. The timing of exchanges can vary, based on criteria e.g. after a specified number of local iterations or after the observation of an improved solution. Stützle (1998) extends the work presented by Bullnheimer et al. (1997d) by considering alternative schemes of speeding up runs of ACO algorithms that use local search algorithms. He proposed the exploitation of parallelisation to experiment with different algorithmic parameter settings. The success (with respect to solution quality) of any meta-heuristic is heavily influenced by the parameters used. Consequently, employing a system that tests a range of experimental values will potentially remove the concern of selecting inappropriate parameter settings.

In a similar vein, Michel and Middendorf (1999) detailed the island model when tackling the Shortest Common Supersequence Problem. The *island model* allowed colonies to work independently for some time until an exchange of best solutions is performed. For each colony, the associated trails are updated according to the feedback function used. The *island model* is a variant of elitism as proposed by Dorigo et al. (1991) and detailed in Bullnheimer et al. (1999).

2.4.3 Ant Colony Optimisation (ACO)

ACO represents a family of ant-based algorithms. The development of more sophisticated algorithms to become suitable and adaptable to a variety of problem types required a more general classification. In addition, the use of local search to enhance solution quality has given rise to a range of hybrid methods.

Ant-Q

Gambardella and Dorigo (1995) introduced Ant-Q, which strengthens the connection between Reinforcement Learning, Q Learning in particular, and Ant Systems. Watkins (1989) introduced the method of reinforcement learning called Q Learning, which is defined as follows. An agent exists within a world that can be modelled as a Markov Decision Process. It observes discrete states and executes discrete actions. Each discrete time step, an agent observes a state x, takes action and observes a new state y and receives reward r. Transitions are made probabilistically according to suitable distributions.

Ant-Q differed from AS in three ways.

1. The action choice rule

The Ant System used the random proportional rule. However, Ant-Q introduced the pseudo-random proportional rule, which combined the AS choice rule with the pseudo-random rule in Q-learning. The pseudo-random proportional rule works as follows: at some vertex insertion decision point, there is a probability q_0 that a vertex is chosen deterministically and the random-proportional rule is used with probability (1-q_0), with $q_0 \in [0,1]$.

2. Type of global reinforcement

The best performing ant (all ants contribute to feedback in AS) in the current iteration contributes to global reinforcement. The reward function is proportional to the length of the tour (with respect to TSP).

3. Trail reinforcement

The trail update rule is as follows:

$$AQ(r,s) = (1-\alpha).AQ(r,s) + \alpha\left\{\Delta AQ(r,s) + \lambda.Max_{z \in J_k(s)} AQ(s,z)\right\} \quad (2.21)$$

$AQ(r,s)$ refers to the Ant-Q trail levels present on the edge (r,s) and $J_k(s)$ is a list of the cities to be visited from city s for the k^{th} ant. Parameters α and λ represent the learning step and the discount factor respectively. The reinforcement term ΔAQ is always zero except after each ant has completed its tour (global update). This (delayed) term is the inverse of the best solution, weighted by some constant. The discount factor $\lambda.Max\ AQ(s,z)$ is greater than zero at each local update stage (i.e. trail updates when an ant is constructing a solution).

Dorigo and Gambardella (1996) quantify the dimension of the solution space that is covered by the ants by the use of the λ-branching factor of vertex r, which is defined as

follows. Let $AQ_{max}(r,s)$ and $AQ_{min}(r,s)$ be the largest and smallest trail values respectively on all edges connecting an exiting vertex r and potential vertex s ($s \in S$, where the set S contains all the vertices that are obtainable after the exiting vertex r) and let $\delta_r = AQ_{max}(r,s) - AQ_{min}(r,s)$. For ϕ, $0 \leq \phi \leq 1$, the λ-branching factor of vertex r is represented by the number of vertices exiting from r with a trail (AQ) value greater than $\phi.\delta_r + AQ_{min}(r,s)$. The lower the λ-branching factor, the nearer the search is to stagnation, Stützle and Hoos (1996), and consequently, the ants concentrate their efforts in smaller areas of the solution space. The authors claimed that a near stagnation environment is desirable since the ants' exploit the high concentrates of pheromone on certain trails and this leads to the construction of solutions that are built around a core group of solution characteristics. Stützle and Hoos (1996) use the λ-branching factor to determine whether local search should be applied to ant-based solutions. Their rule was as follows. When $\lambda < 2$, local search should be used on all solutions since best achievable solutions are found during a near-stagnation environment. When $\lambda \geq 2$, local search is applied to selected solutions.

Ant Colony System (ACS)

ACS, Dorigo and Gambardella (1997), superseded Ant-Q and differs by one key factor. The discount factor in equation 2.21 is replaced by a constant, defined as τ_0, due to the insignificant differences in solution quality. ACS is preferred to Ant-Q due to its relative simplicity.

Min Max Ant System (MMAS)

Stützle and Hoos (1996) introduced the concept of MMAS, which can be seen as a more rigorous, direct and quicker search than AS. MMAS allows a population of ants to construct solutions but only the best ant is permitted to update the trails after each iteration on either a local or global basis. This system is a greedier version of AS and consequently, some trail levels could become exorbitant and lead the process into stagnation. To help prevent this unwanted occurrence, Stützle and Hoos imposed trail limits. An upper trail limit, τ_{max}, prevents large trails, while the lower trail limit τ_{min}, encourages the selection of less commonly chosen solution characteristics. Each trail strength is limited to the interval $[\tau_{min}, \tau_{max}]$.

The value τ_{max} is defined as a geometric series and is calculated as follows:

$$\tau_{\max} = \frac{1}{1-\rho} \cdot \frac{1}{f(s)} \quad (2.22)$$

where ρ is the rate of evaporation and $f(s)$ refers to the fitness of the global-best solution. The parameter τ_{max} is modified according to the conditions of the run. At the start of the run experiment, τ_{max} is set at some arbitrary high value to ensure that, after the completion of the first iteration, all trail levels adhere to τ_{max}.

Stützle and Hoos indicated that the determination of suitable lower trail limits could be quantified (theoretically) through monitoring the convergence of the search. They claimed that a run of the MMAS has converged if the best solution is constructed with a probability significantly higher than *0*, which was defined as p_{best}. They also stated that when stagnation occurs, the trails associated with solution characteristics (edges in TSP for example) would correspond to the upper trail limit τ_{max} and other solution components will be approximately equal to τ_{min}.

The authors made a series of assumptions and simplifications to arrive at a theoretical value of τ_{min}. They stated reasonably that when an ant constructs the best found solution, it is required to make, at each decision point, the 'right' decision. Each decision depends on the distribution of trails, since the bias towards some decisions will be stronger than others and also will be indirectly determined by the lower and upper trail limits. The authors assume though that these decisions (making the right choice) are constant across all decisions that the ants have to make. This probability is defined as p_{dec}. The probability that an ant constructs the best solution again is $p_{dec}^{(n-1)}$, as an ant has to make *n-1* decisions to construct a solution. By equating

$$p_{dec}^{(n-1)} = p_{best} \quad (2.23)$$

then p_{dec} is defined as

$$p_{dec} = \sqrt[n-1]{p_{best}} \quad (2.24)$$

It was assumed that the ants have to choose between *n/2* cities and on average among *avg* cities, which yields

$$\frac{\tau_{\max}}{\tau_{\max} + avg.\tau_{\min}} = p_{dec} \quad (2.25)$$

After rearrangement we have

$$\tau_{\min} = \frac{\tau_{\max}.\left(1 - p^{dec}\right)}{avg.p^{dec}} \quad (2.26)$$

To avoid the sensitivities of parameter settings, Stützle and Hoos (1996) suggested the *smoothing of trails*, which increases the trails proportionally to the difference between τ_{ij} and τ_{max} if stagnation of the algorithm is detected.

Stützle and Hoos (1996) detailed the following *smoothing of trails* strategies:

Proportional update: the trail intensity on each edge is increased by a proportion of the difference between the upper trail limit and the current trail intensity.

Lift minimum: the lower trail limit is increased by some measure and those trails below the updated τ_{min} are set to τ_{min}.

While Stützle and Hoos (1997) acknowledged the potential of these philosophies, it was observed that setting all trails to τ_{max} achieved the best results (when stagnation is detected). This strategy reinitialises the feedback passed to the trails and achieves the same as re-starting the ant searches.

At each update phase, the iteration-best or global-best ant updates the trails with the reward that is proportional to the fitness of the solution. The authors observed that the use of the iteration-best ant would allow greater exploration, while the global-best ant will encourage early stagnation.

Fast Ant System (FANT)

Taillard and Gambardella (1997) proposed another ACO type algorithm for the Quadratic Assignment Problem (see below). FANT uses only one ant per cycle with no heuristic

information and aims to achieve good solutions quickly. The trails are not evaporated and are updated as follows.

$$\tau_{ij}(t+1) = \tau_{ij}(t) + r.\Delta\tau_{ij} + r^{*}.\Delta\tau_{ij}^{gb} \quad (2.27)$$

The parameters r and r^* refer to the relative importance of the current and global-best solution. Additionally, we set $\Delta\tau_{ij}=1$ or $\Delta\tau_{ij}^{gb}=1$ for each (i,j) component of the current and global best solutions respectively. An ant search system that uses only one ant per cycle will be more susceptible to stagnation due to the inherent lack of variation that exists with limited ant searches. Consequently, the use of *Intensification* and *Diversification* strategies play decisive roles in the success (with respect to solution quality). These search aids are managed through the dynamic variation of the weight r. Typically, after the return of a new global best solution, the relative weight of the current solution is set to 1 and all pheromone trails are reinitialised. The authors claim that this action has the effect of intensifying the search in the neighbourhood around the new global best solution. Also, when lack of exploration is detected, more weight is placed on r, which encourages greater exploration. To enhance solution quality further, local search is applied to all generated solutions.

2.4.4 Major Applications

This section will demonstrate the flexibility of Ant Algorithms through reviewing a range of literature of real-life practical applications.

University Course Timetabling Problem (UCTP)

The university course timetabling problem is the closest relative to the examination timetabling problem. It is only recently that ant-based UCTP related literature has appeared, indicating the infancy of research in ant-related university timetabling. A typical feasible timetable is one that satisfies the following hard constraints.

1. No student or lecturer can attend more than one event as the same time.
2. No room can host two lectures at the same time.
3. Seating capacities must be observed.

Typical soft constraints are

1. Minimising the number of students that have two lectures in a row.
2. Minimising the number of students that have just one lecture in a day.

The UCTP is described graphically. The vertices in the graph represent the timeslots and edges joining two vertices indicate that the two courses cannot be allocated to the same timeslot. Socha et al. (2002) applied the MMAS to the UCTP. The authors address two pheromone matrix representations (used as two separate approaches). The first trail considers the absolute position of events. Trails are stored in *events* by *timeslots* matrix. The second trail regards the relative position of events and measures the value of placing or not placing certain events together. Trails are stored in *events* by *events* matrix, which mirrors the representation presented by Costa and Hertz (1997). It is the second trail that the authors claim that has the greater potential to succeed. The authors argue that, given a perfect timetable, many permutations of timetables can be obtained without affecting overall solution quality, thus the absolute position of events is not critical but relative. It is claimed that quicker learning is evident through this style of trail representation. Despite favouring the mechanics of the second representation, results indicate that the first type of representation performs significantly better.

In Socha et al. (2002), the usual MMAS conditions apply. Only the best ant contributes to trail level updates and no heuristic information is used in favour of local search. Trails $[\tau_{min}, \tau_{max}]$ were limited to $[0.0078, 3.3]$ for smaller data sets and $[0.0019, 3.3]$ for medium and large data sets. Universal parameter settings are applied for the following: $\rho=0.30$, $\alpha=1$ and the number of ants is fixed at 10. There is no β parameter due to the absence of heuristic information (feature of MMAS in this instance).

A further ant-related study was presented in Socha et al. (2003). This paper discusses the differences in application and effectiveness of MMAS and ACS. These algorithms differ in the manner in which heuristic information, update procedures (both global and local updates performed for ACS, with only global for MMAS) and local search is utilised. Differences in parameter settings do exist.

For ACS, $\alpha=0.1$, $\rho=0.1$ and the number of ants per iteration is set to 10. The global pheromone update rule is as follows.

$$\tau(e,t)=\begin{cases}(1-\rho).\tau(e,t)+\rho.\dfrac{g}{1+q(Global\,best)} & if\ (e,t)\ is\ in\ global\ best \\ (1-\rho).\tau(e,t) & otherwise\end{cases} \quad (2.28)$$

The parameter q represents the number of constraint violations with respect to the global best solution and g is a scaling factor that was given a value of 10^{10}.

The local update equation follows the form that has been described previously in this Chapter.

For MMAS, the global update rule is as follows.

$$\tau(e,t)=\begin{cases}(1-\rho).\tau(e,t)+1 & if\ (e,t)\ is\ in\ global\ best \\ (1-\rho).\tau(e,t) & otherwise\end{cases} \quad (2.29)$$

The same trail level restrictions (the interval $[\tau_{min}, \tau_{max}]$) are imposed as in Socha et al. (2002)

For both MMAS and ACS, trail is stored in an *events* by *timeslots* matrix.

It is demonstrated that MMAS is better in all problem instances and does particularly well on large problems in comparison to established methods. It can be seen that the key factor in result disparity between MMAS and ACS is the use of local search. With MMAS, the solution that causes the fewest number of constraint violations is selected for improvement through local search (10,000,000 step allowance). With ACS, all ants are subjected to local search improvement. A two-phase strategy is used here. If the current iteration is less than a pre-specified value j (set to 11) then s_1 (50,000) local search steps are allowed, otherwise s_2 (20,000) steps are permitted. A lower step size s_2 suggests that less timetable repair is required at latter stages of the ant search given that solution quality improves (until some dampening point) as the search matures. The authors also indicated that the gulf in solution

quality could be narrowed through modifying ACS. Heuristic information[1] is removed and the number of timetables that is repaired is reduced. Solution quality is improved to the extent that ACS competes with MMAS.

Quadratic Assignment Problem

For the QAP, we are given n locations and n facilities and each facility is required to be assigned to a location. There are given distances between the locations and given flows between the facilities. The objective is to generate a location-facility assignment in such a way that the sum of the product between flows and distances is minimal. Mathematically the problem is defined by three matrices of dimension n x n.

D=[d_{ih}], which stores the distances between locations i and h.
F=[f_{jk}], which stores the flows between activities j and k.
C=[c_{ij}], which stores the assignment costs of activity j to location i.

Ahuja et al. (2000) discussed the QAP for the Campus Planning Problem and this application will act as an illustration of the QAP in practice. There are new facilities to be constructed on campus and the objective is to minimise the walking distance for staff and students. We suppose that there are n available sites and n facilities to locate. With respect to the above formulation, we define d_{ih} as the walking distance from sites i and h, while f_{jk} represents the number of people per week that travel between facilities j and k. A cost c_{ij} is incurred if there is some on-site fixed cost of locating facility j at site i.

The problem is formulated to accommodate a matrix X that is dimensioned by activities and locations. An element $x_{ij}=1$ indicates that activity j is assigned to location i.

The objective function now becomes.

$$Min \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{h=1}^{n}\sum_{k=1}^{n} d_{ih}.f_{jk}.x_{ij}.x_{hk} + \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} \quad (2.30)$$

Subject to the constraints

[1] Heuristic information represented the number of constraint violations caused by the insertion of a course within the partial solution. Parameters β (3.0) and γ (2.0) weighted the hard and soft constraints respectively.

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \forall j = 1,...,n \quad (2.31)$$

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \forall i = 1,...,n \quad (2.32)$$

$$x_{ij} = 0 \text{ or } x_{ij} = 1 \quad \forall i=1,\ldots,n \quad \forall j=1,\ldots,n \quad (2.33)$$

AS was initially applied to the TSP and the QAP is a generalization of the TSP, hence the extension of AS to the QAP was a natural progression. Maniezzo et al. (1994) first applied AS to the QAP (AS-QAP). Facility i was assigned to location j on a probabilistic basis biased according to the product of pheromone trail and desirability. The latter is defined as $n_{ij}=1/e_{ij}$, where $e_{ij}=f_i.d_j$, which represents the product of the distance potential of location i and flow potential of facility j. Two vectors d and f are calculated in which the i^{th} member represents respectively the sum of the distances between location i to all other locations and the sum of the flows between facility i to all other facilities. The AS-QAP uses the random proportional rule and for all location-facility *couplings*, the pheromone update rule is as follows:

$$\tau_{ij}(t+1) = \rho.\tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}{}^{k} \quad (2.34)$$

Parameters within the above expression have been defined in Chapter 1. $\Delta\tau_{ij}{}^{k}$ is the amount of pheromone that ant k deposits on coupling i,j and can be defined as:

$$\Delta\tau_{ij}{}^{k} = \frac{Q}{J_{\psi}{}^{k}} \quad (2.35)$$

where $J_{\psi}{}^{k}$ refers to the fitness value of the k^{th} ant solution and Q is the constant amount of pheromone deposited by the ant.

An improvement to the AS-QAP was presented in Maniezzo and Colorni (1999), defined as AS2-QAP. The key differences between these approaches are as follows.

The action choice rule is transformed into a linear function. Given a location j, ant k, the probability that facility i is assigned to this location is calculated through

$$p_{ij}^{\ k} = \frac{\alpha.\tau_{ij}(t) + (1-\alpha).\eta_{ij}}{\sum_{r \in N_r^k} \alpha.\tau_{ir} + (1-\alpha).\eta_{ir}} \quad if\, i \in N_j^k \quad (2.36)$$

where N_j refers to the list of facilities that can be assigned to location j.

This action rule was introduced to make the search process more efficient. The use of bias exponents in this function is expensive with respect to runtimes. This expense is magnified by the frequency of action rule usage since the majority of all solution insertions make reference to this probabilistic function. This topic will be investigated in greater depth in Chapter 3.

The desirability term in the above action rule was based on the Gilmore and Lawler bound (GLB). This bounding strategy evaluates the value of a completed assignment after the allocation of facility i to location j. Maniezzo (1999) also used a lower bound strategy to evaluate the potential of an assignment when applying the Approximate Non-Deterministic Tree Search (ANTS) to the QAP. However, the GLB in Maniezzo (1999) was deemed computationally expensive and consequently, lead to the conception of a more efficient, but weaker lower bound, known as the LBD. It was shown that the LBD was good enough to guide the ants' solution construction.

Both Maniezzo (1999) and Maniezzo and Colorni (1999) tackled the issue of stagnation. Trail update was based on a linear dynamic scaling function, Equation (2.37). After k solutions are available, a moving average z_{AVER} is computed. Each new solution z_{CURR} is compared to z_{AVER}. If $z_{CURR} < z_{AVER}$, the trail level of the last solution's moves is increased, otherwise it is decreased.

$$\Delta\tau_{ij} = \tau_0 \cdot \left(1 - \frac{z_{CURR} - LB}{z_{AVER} - LB}\right) \quad (2.37)$$

LB is a lower bound to the optimal solution cost and τ_0 is a constant. Maniezzo states that the use of a dynamic scaling procedure penalizes small achievement in the latest stage of

the search, while avoiding a narrow search process after the observation of good solutions during the early phase of the search.

An alternative approach to the QAP, HAS-QAP (Hybrid Ant System for the QAP), was presented in Gambardella et al. (1999b). The pheromone trail is not used to construct solutions, but rather to modify random starting solutions through solution swaps. In a solution Π, an index r is chosen on a random basis, along with a second index $s \neq r$ and the elements Π_r and Π_s are swapped. Index s is selected deterministically with a probability q_0 or stochastically with a probability $(1- q_0)$ according to the following rule.

$$\frac{\tau_{r\Pi_s} + \tau_{s\Pi_r}}{\sum_{j \neq r} \left(\tau_{r\Pi_j} + \tau_{j\Pi_r} \right)} \quad (2.38)$$

Each solution is further enhanced through the use of local search. Element swaps that generate improved fitness are accepted. HAS-QAP allows for *Intensification* and *Diversification* strategies. The former is managed as follows. After a new best solution is obtained, the intensification mechanism is activated. Each ant starts its iteration with the best permutation found and a search is performed in the neighbourhood where the new best solution is found. Meanwhile, the latter is managed as follows. If no improvement has been witnessed for S iterations then the pheromone trails are re-initialised.

In the spirit of ACS, only the best solution updates the trails at the end of each cycle. The reward function quantifies the quality of the solution obtained.

Job-Shop Scheduling (JS)

We have a set $M=\{M_1, \ldots, M_m\}$ of machines, a set $J=\{J_1, \ldots, J_n\}$ of jobs and a set of operations $O=\{\mu_{ij}\}$, $(i,j) \in I$, where $I \subseteq [1,n]x[1,m]$. For an operation $\mu_{ij} \in O$, there is a job J_i to which it belongs, a machine M_j on which it has to be processed and a processing time p_{ij}. The objective is to create a schedule specifying when each task is to begin and what resources it will use that satisfies all the constraints while taking as little time as possible. The JS can be described graphically where each vertex represents the processing of a job by a machine. An extra vertex, labelled v_0, is added to enable the specification of job

ordering for a particular machine. The weight p_{ij} is associated to each vertex, a directed edge represents precedence relations and undirected edges represent machine constraints.

Colorni et al. (1994) suggested that AS could be used to solve JS. All ants begin at v_0 and construct their permutations based on the balance of trail and desirability. Once a job is added to the ordering it is appended to a tabu list to prevent revisiting already scheduled jobs.

The discussion in Colorni et al. (1994) was concentrated on parameter sensitivity (N_A, α, β and ρ) for a range of JS problem types. It was shown that setting N_A to the number of jobs, α=1, β=1 and ρ=0.7 were the optimum settings. It was stated that the best AS results were, on average, *10%* away from the known optimums. Tests were performed on four different problems.

Frequency Assignment Problem (FAP)

The task in the FAP for cellular telephone networks is to allocate carrier frequencies to base stations in such a way that a possibly large amount of telephone traffic is supported and simultaneously a good quality of connections is guaranteed. This problem can be modelled as a generalised graph colouring problem (giving obvious parallels with ANTCOL). Vertices represent the connections and edges between the connection vertices indicate some minimal distance between the frequencies, as stored in a Channel Separation Matrix (CSM) for all pairwise combinations of connections. Maniezzo and Carbonaro (2000) applied the ANTS heuristic to the FAP. The framework of this study was based on the work presented by Maniezzo (1999). Please refer to the ANTS heuristic for QAP for methodology. The ANTS strategy was compared with DSatur, TS and SA and the results presented were very competitive. Of the 35 data sets used for experimentation, 23 of them were the best-known results. Additionally, such solutions can be achieved in relatively efficient runtimes.

Sequential Ordering Problem (SOP)

The main objective of the SOP is to obtain a sequence of jobs that minimises the total makespan time subject to precedence constraints. The SOP can be formulated as an Asymmetric Travelling Salesman Problem (ATSP) with the cities representing the jobs and

the distance between the cities symbolising the waiting time between the end of one job and the start of another. A tour must begin at *city 0* and end at *city n*, while visiting all the intermediate cities once in an order that minimises total distance. Gambardella and Dorigo (2000) introduced the HAS-SOP, which combines the construction methodology of ACS and a Local Search procedure. The trail related issues (e.g. update, reward) are the same as presented before. Local Search carries each ant-based solution towards the local minimum and achieves this by exchanging edges. A *k-exchange* deletes *k* edges from a solution creating *k disjointed paths* that are reconnected by *k* new edges. The authors discussed the *lexicographic path preserving edge exchange*, which only allows feasible edge exchanges (subject to precedence constraints). Comparative analysis with alternative methods proved that HAS-SOP is very competitive. Thirteen *small* (=100 vertices) problems were tested and it was shown that HAS-SOP returns the optimal solution on nine occasions. Nine *big* (>100 vertices) problems were used for experimentation, with optimal solutions being found for three of them.

Shortest Common Supersequence Problem (SCS)

The official definition of the SCS is as follows. Given a finite set L of strings over an alphabet Σ, find a string of minimal length that is a supersequence of each string in L. For example, consider the alphabet $\Sigma=\{a, b, c\}$ and the set of strings $L=\{cbbc, abc, cba\}$. It can be seen that the shortest common supersequence is *cbabc*.

	c	**b**	**a**	**b**	**c**
cbbc	X	X	-	X	X
abc	-	-	X	X	X
cba	X	X	X	-	-

Strings can represent various sequences of commands. An example is the genetic information of living creatures, which is stored in large DNA molecules and can be modelled as a string.

Michel and Middendorf (1999) introduced the AS-SCS heuristic. For the i^{th} position in the string, an ant makes a decision of the symbol to allocate based on a balance between

pheromone and desirability. AS-SCS differs from AS in the sense that it uses a *lookahead* function that evaluates the symbol that is associated with the maximum trail level in the succeeding state. This measure acts as the desirability η. The Majority Merge (MM) heuristic is a traditional heuristic for the shortest supersequence problem. Its role within AS-SCS is to weight pheromone values in order to guide ants towards good solutions. AS-SCS with lookahead and the incorporation of the MM heuristic leads to new notation – AS-SCS-LM.

Ant System for the Vehicle Routing Problem (VRP)

The VRP is an extended version of the TSP. The VRP can be represented by a weighted undirected graph $G=(V,E,d)$ where $V=\{v_0, v_1, v_2,...,v_n\}$ is a set of vertices and $E=\{v_i, v_j: i\neq j\}$ is a set of edges. The vertex v_0 symbolises the depot, while the other vertices represent the cities/customers. A weight d_{ij} on an edge that connects v_i and v_j quantifies the distance between the two cities. Additionally, customer v_i is equipped with a demand q_i and a service time δ_i. The main objective of the VRP is to find an allocation of vehicles to routes while minimising costs. Additionally, each customer is only visited once and all vehicles begin and end at the depot. Routes are also subjected to total length and vehicle capacities.

Solutions are constructed by successively choosing cities to visit. A new tour is started (from the depot) when the selection of another city would lead to an infeasible solution (through capacity violation).

Bullnheimer et al. (1997b) recognised that the inclusion of savings and capacity utilisations would enhance solution quality. However, due to the computation efficiency concerns, these characteristics were incorporated within the desirability function, defined as the *parametrical saving function*. Let f represent the savings and g, the capacity utilisation. The desirability is defined as:

$$\eta_{ij} = d_{i0} + d_{0j} - g.d_{ij} + f.\ /d_{i0} - d_{0j}/ \quad (2.39)$$

Meanwhile, Bullnheimer (1997c) reverted to the desirability function as suggested by Dorigo et al. (1991) and incorporated savings and capacity functions for the VRP. Savings measure the suitability of including two cities v_i and v_j in a tour and is quantified as follows:

$$\mu_{ij} = d_{i0} + d_{0j} - d_{ij} \quad (2.40)$$

High μ_{ij} indicates that visiting v_j after v_i is a good choice.

A capacity term, κ_{ij}, encourages the high utilisation of vehicles.

$$\kappa_{ij} = \frac{(Q_i + Q_j)}{Q} \quad (2.41)$$

Where Q_i – total capacity used, including customer v_i

Q_j – utilisation of customer v_j

Q – total capacity on route

To accommodate these terms, the random proportional rule becomes:

$$p_{ij} = \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}[\mu_{ij}]^{\gamma}[\kappa_{ij}]^{\lambda}}{\sum_{h \in \Omega}[\tau_{ih}]^{\alpha}[\eta_{ih}]^{\beta}[\mu_{ih}]^{\gamma}[\kappa_{ih}]^{\lambda}} \; if j \in \Omega \quad (2.42)$$

where the set Ω contains all the cities that can be visited after city i. Both Bullnheimer (1997b) and Bullnheimer (1997c) incorporated elitism. Bullnheimer (1997c) reduced tour lengths further through a 2-opt Sweep Local Search algorithm (which is not detailed here).

Multiple Ant Colony System for VRP with Time Windows (MACS-VRPTW)

VRPTW differs from VRP by its objective function. VRPTW minimises the number of vehicles and then the total travel time. Gambardella et al. (1999a) detailed the MACS-VRPTW, which uses two near independent ACS based colonies, labelled ACS-VEI and ACS-TIME. The former (major objective) attempts to lower the number of vehicles while the latter minimises vehicle movements.

When a global best solution of k vehicles is obtained through ACS-VEI, ACS-TIME constructs tours that use k vehicles while minimising the total time required to visit all the

customers. Meanwhile, ACS-VEI searches for a solution that serves all customers with *k-1* vehicles. Once an improved solution is observed (i.e. one with *k-1* vehicles) in ACS-VEI, ACS-VEI and ACS-TIME are reinitialised and two new colonies are activated that attempt to solve the problem with respect to the reduced number of vehicles.

At the end of each cycle, the global best solution is used to update each trail and the infeasible solution with the highest number of visited customers is also used to update ACS-VEI.

At each customer-decision stage, the desirability η_{ij} is computed by taking into account the travelling time t_{ij} and the number of times a customer j has not been inserted into solutions, thus favouring the customers that have had less prior involvement.

A similar study was presented in Doerner et al. (2001) when an ACO approach is proposed for minimising total costs in a transportation network. Two trails were again used. The first (master) minimises fleet size, while the second (slave) minimises vehicle movement costs. In Gambardella et al. (1999b), only the global best ant contributes towards pheromone levels, whereas in Doerner et al. (2001) the top a ants contribute to each trail. Good solutions are injected solely from the slave into the master population. Therefore, there is more cooperation between the colonies in this instance in comparison to MACS-VRPTW.

2.4.5 Less Relevant Studies

Ant Algorithms have been applied to non-combinatorial optimisation problems. Examples of such studies are as follows.

1. Ant Colony Optimisation for Virtual-Wavelength-Path Routing and Wavelength Allocation – Varela and Sinclair (1999)
2. Distributed Stigmergetic Control for Communications Networks – Di Caro and Dorigo (1998)
3. Load Balancing in Tele-Communication Networks – Schoonderwoerd et al. (1994)

Therefore, it is evident that ant algorithms are becoming increasingly popular by their application to problem types other than combinatorial optimisation.

2.5 Conclusion

This Chapter has examined a cross section of literature that is relevant to the investigation within this thesis. Copious amounts of examination scheduling literature exist and some date back to the 1960's. Approaches vary from basic heuristics, tailor made heuristics and off-the-peg techniques. However, the latter class is rare due to the different nature of the problem at different institutions, see Burke et al. (1995a). However, it should be stated here that exact methods are not viable options due to the computational difficulty (classified as NP-Hard) of the examination timetable.

Many pieces of examination literature propose heuristic techniques that utilise single phase or multi phase methods. The former tackles the problem in one sweep of the algorithm and does not use a lookahead facility. Consequently, large costs may be encountered towards the latter end of construction. The latter solves the problem in stages, with each stage focusing on one objective. Solution quality will potentially increase as a result, however, decisions made in earlier phases remain fixed for latter stages and may lead to poorer solutions than desired. These drawbacks led to the use of improvement strategies to enhance solution quality. Local search and TSP have played prominent roles here. Simulated Annealing is a more sophisticated variant of local search and has been applied successfully recently to examination scheduling. The same can be said of Tabu Search.

Recently, research has been carried out in the field of evolutionary algorithms. The last decade has observed the onset of genetic algorithms as a scheduling tool. It has been demonstrated by a series of authors that the basic form of GA's is not appropriate, but that hybridisation with local search offers much potential. The need for more research is clear.

It has been shown that various applications of Ant Algorithms to real-life practices have been very encouraging and, in some cases, results outstrip previous benchmark solutions. It has been noted that this search technique has not been applied to examination scheduling and consequently, leaves room for research and forms the basis of this thesis. However, it

should be noted that MMAS (and ACS to an extent) has been applied to the course timetabling problem.

The relevance of the graph colouring problem can be seen through the numerous applications to the examination timetabling problem. Section 2.2 showed that graph colouring was the underlying model of ANTCOL and consequently, plays a pivotal role within this thesis.

Vesel and Zerovnik (2000) argue that the ANTCOL algorithm is not competitive as a graph colouring solution method when placed in comparison with the Petford-Welsh algorithm, Petford and Welsh (1989). The authors claim that a simple repeated RLF for larger graphs is also superior to ANTCOL. However, Costa and Hertz (1997) does not suggest that ANTCOL is the ultimate graph colouring method since they acknowledge the marginal superiority of an alternative method, a GA-Steepest Descent hybrid, Costa et al. (1995). Additionally, it should be clarified that the ANTCOL algorithm is used merely as the raw framework of an examination timetabling solution method. The frequent use of graph colouring heuristics as examination timetabling methods suggest that the ANTCOL variant is a wise starting point for the task at hand.

This literature survey has shown that wide-ranging solution methods have been used. The examination timetabling problem is still open to further research and the intention here is to assess the suitability of a relatively new evolutionary heuristic as a general solution method.

Chapter 3

Precursory Investigation

3.1 Introduction

Chapter 1 discussed the ANTCOL algorithm as introduced by Costa and Hertz (1997). In this chapter we gain insight into the mechanics of the procedure. The components of the algorithm are analysed and conclusions regarding appropriate parameter settings and the efficiency of the method are drawn. The results generated through this investigation will provide the basis for future work.

The structure of this chapter is as follows. Firstly, the data sets are introduced and then we examine the basic mechanics of the algorithm, which encompasses the influence of the various parameter settings and the benefit of using trail accumulation over the basic single pass heuristic. We then note the advantage of using a random proportional rule (RPR) in contrast to a deterministic rule. Due to the computational expense of the ant algorithm, some speed-saving implementations are proposed to increase the attractiveness of the algorithm and to allow for a greater number of experiments.

3.1.1 Data Sets

This thesis will use a group of twelve data sets, which are publicly available from the Internet (ftp://ftp.mie.utoronto.ca/pub/carter/testprob/). These data are used in order to make cross comparison possible to assess the suitability of the ant-based method. However, these data do not come accompanied with the side constraints, such as pre-assigned and time-windowed exams, usually associated with the examination scheduling problem. To compensate, two real-life data sets are added to the 'data pool', namely Swan2000 and Swan2002, which as the dataset names suggest originate from the University of Wales Swansea. These data are accompanied by some side constraints and maximum seating capacities. All fourteen data sets are described below.

Code	Location	Abbrev.	Exams	Students	Enrolments	Sessions	Density
Car-s-91	Carleton University	C91	682	16925	56877	35	12.8%
Car-f-92	Carleton University	C92	543	18419	55522	32	13.8%
*Ear-f-83**	Earl Haig Collegiate	EAR	189	1125	8109	24	26.7%
*Hec-s-92**	Ecole des Hautes	HEC	80	2823	10632	18	42.0%
Kfu-s-93	King Fahd University	KFU	461	5349	25113	20	5.6%
Lse-f-91	London School of Economics	LSE	381	2726	10918	18	6.3%
Rye-f-92	Ryerson University	RYE	486	11483	45051	23	7.5%
Sta-f- 83	St Andrews	STA	139	611	5751	13	14.4%
*Tre-s-92**	Trent University	TRENT	261	4360	14901	23	5.8%
Uta-s-92	Toronto University Arts&Science	UTA	622	21266	58979	35	12.6%
Ute-s-92	Toronto University Engineering	UTE	184	2750	11793	10	8.5%
Yor-f-92	York Mills Collegiate	YOR	181	941	6034	21	28.9%
Swan2000	Swansea University	Swan2000	313	4611	15857	20	10.33%
Swan2002	Swansea University	Swan2002	722	6388	31094	36	4.20%

Table 3.1 – Data set information

Individual investigations do not utilise the full data pool since this would involve large computational effort and consequently, attention is consistently paid to a core of three data sets (marked by single asterisk) to enable parameter calibration. The conclusions drawn from these studies are then applied universally to the other data sets within the data pool.

HEC was chosen because it is arguably the most difficult data set since it has the highest density (defined below) and the size of its largest clique (defined below), 17, is near to the number of timeslots, 18. EAR was selected given that there are many maximum cliques (37 of 22 vertices), thus increasing colouring difficulties. Meanwhile, TRENT has a lower density but has a greater number of exams and consequently, this allows insight into the computational efficiency of methods.

The density of a graph can be computed as follows.

$$\frac{total\ number\ of\ edges}{n(n-1)} \times 100 \quad (3.1)$$

where n represents the number of vertices

A clique relates to a subset of vertices that form a complete subgraph (all pairs of vertices are joined by an edge) where each vertex requires a distinct colour. Therefore, the maximum clique is often used as a lower bound in colouring problems. However, this thesis will often refer to benchmark results for evaluation of proposed methods.

3.1.2 Computer Facilities

The attractiveness of solution quality will depend not only on the perceived fitness of the timetable but also on the speed of computation. There is no real outstanding benefit in achieving a very good solution that requires exorbitant runtimes to achieve. Computational effort is recorded in seconds.

Due to the volume of computations and storage required for Ant Algorithms, it is possible that inefficient use will lead to excessive runtimes. Therefore, much focus is placed upon computational effort throughout this thesis.

Work has been performed on Pentium III Mertec Systems. Due to the volume of experiments that have been performed, the use of more than one terminal was necessary.

Programs were initially written in Visual Basic 6 and then were converted to Salford Fortran 95 to improve runspeeds. It was observed that runspeed could be quickened further through the use of the *Optimiser* (to maximise efficiency of object code) facility. Only executables were used to run the experiments. It is possible to use any machine (even if Fortran 95 is not installed) that has the *salflibc.dll* system file within the appropriate folder.

Five independent runs were performed for each experiment.

3.2 Basic Mechanics of the Algorithm

In this section, we examine the basic elements of the algorithm – sensitivity of solution quality through changes in bias parameters, the number of ants per cycle and the number of cycles.

3.2.1 Graph Colouring Heuristics

Initially, solutions were constructed using RLF and DSatur based graph colouring heuristics, as presented in Chapter 2. These results allow for future comparison and indicate whether the use of trail offers any significant benefit to solution quality. The results are tabulated below.

Av – average numbers of colours across five experimental runs.
Best – minimum number of colours across five experimental runs.

	HEC		EAR		TRENT	
Construction Heuristic	*Av*	*Best*	*Av*	*Best*	*Av*	*Best*
A	21.74	18	32.19	27	28.34	24
B	22.63	19	33.61	28	29.92	26
C	22.02	19	32.47	28	28.71	24
D	20.31	18	28.29	25	26.16	23
E	20.87	18	29.20	26	27.41	24
F	20.45	18	28.61	25	26.50	23
G	21.38	18	30.94	27	27.84	24
H	22.40	19	33.37	28	30.57	26

Table 3.2 – Statistics describing Single Pass Heuristics

There is a distinctive trend with regards to the performance of the construction heuristics. Ranking the average scores shows that the construction heuristics perform similarly across the data sets (Appendix 3.1). Ranks 1-6 correspond to heuristics D, F, E, G, A and C respectively, with rank 1 relating to the lowest average score. Meanwhile, heuristic H is ranked 7 for HEC, 7 for EAR and 8 for TRENT and heuristic B is ranked 8, 8 and 7. Note that construction heuristics D, E and F occupy the top 3 ranks. These heuristics are Recursive Largest Degree First with the first vertex in each colour chosen according to the greatest degree in the uncoloured subgraph, while random selection of the first vertex does not fare so kindly.

3.2.2 Influence of Bias Parameters

From this juncture, trail is accumulated to represent the suitability of constructions. As detailed in Chapter 1, the ants construct solutions and the reward function evaluates the fitness of the solution and this acts as a feedback mechanism. Within each colour, the trail between each pairwise exam set is updated according to the level of additional trail (feedback) offered. At a vertex-colour decision stage, an ant makes an allocation biased to

the level of trail between the feasible uncoloured vertex and coloured vertices within a colour group. The RPR is a balance between trail and desirability and bias is represented through parameters α and β respectively. In this section, we examine the sensitivity of solution quality as the bias parameters are varied.

The following discussion uses results that are presented in Appendix 3.2. Increasing β does improve solution quality, but not to the extent that increasing α offers. The row labelled $\alpha=0$ presents the results when no trail is accumulated and only the construction heuristic is used to form solutions. As α increases, there is a notable increase in solution quality. For $\alpha=0.5$ and $\alpha=1$, solutions are superior to $\alpha=0$, however the marked improvement comes with $\alpha\geq2$ (please refer to Table 3.3). For $\alpha=3$ and $\alpha=4$ solution improvement stagnates. Since larger exponents require increased computational effort (as detailed in Section 3.7) the selection of $\alpha=2$ appears to be a sensible choice of this bias parameter. Generally, solution quality does improve as β increases, however, such improvements are fairly minimal for higher settings of α and consequently, this suggests that setting $\beta=1$ should prove adequate.

		Av						Best					
		$\alpha=0$	$\alpha=0.5$	$\alpha=1$	$\alpha=2$	$\alpha=3$	$\alpha=4$	$\alpha=0$	$\alpha=0.5$	$\alpha=1$	$\alpha=2$	$\alpha=3$	$\alpha=4$
HEC	A	21.74	21.21	20.42	18.14	18.18	18.10	18	18	17	17	17	17
	B	22.63	22.29	21.72	19.25	18.81	19.13	19	19	18	17	17	17
	C	22.02	21.22	20.66	18.09	18.19	18.10	19	18	18	17	17	17
	D	20.31	20.13	20.18	19.06	19.81	19.80	18	17	18	18	18	18
	E	20.87	20.81	20.77	21.75	21.58	21.59	18	18	18	18	18	19
	F	20.45	20.24	20.07	20.02	19.06	19.62	18	18	17	18	18	18
	G	21.88	20.93	20.25	18.21	18.38	18.17	18	18	17	17	17	17
	H	22.40	22.14	21.79	21.17	20.92	20.80	19	19	18	17	18	18
		21.54	**21.12**	**20.73**	**19.46**	**19.37**	**19.41**	-	-	-	-	-	-
EAR	A	32.19	31.41	29.81	25.27	25.34	25.00	27	26	25	23	23	23
	B	33.61	33.12	32.03	27.22	26.24	26.20	28	28	28	24	24	23
	C	32.47	31.71	30.13	24.81	24.19	24.97	28	26	25	23	23	23
	D	28.29	27.58	26.12	25.50	25.28	25.08	25	24	24	24	24	24
	E	29.20	28.74	27.84	25.50	25.28	25.08	26	25	25	24	24	24
	F	28.61	27.91	26.31	25.51	25.69	26.24	25	25	24	24	24	24
	G	30.94	29.80	27.96	24.88	24.46	24.13	27	25	24	23	23	23
	H	33.37	33.02	32.52	31.12	30.61	30.48	28	28	28	26	26	25
		31.09	**30.41**	**29.09**	**26.23**	**25.89**	**25.90**	-	-	-	-	-	-
TRENT	A	28.34	27.94	27.24	22.62	22.49	22.49	24	24	24	20	21	21
	B	29.92	29.71	29.18	24.15	22.81	23.28	26	25	25	21	20	21
	C	28.71	28.34	27.65	22.53	22.31	22.14	24	24	24	20	20	20
	D	26.16	25.74	24.40	23.70	23.30	23.48	23	23	21	22	22	22
	E	27.41	27.26	26.64	24.76	24.91	25.45	24	24	23	23	23	23
	F	26.50	26.13	24.97	23.19	23.73	23.84	23	23	21	22	22	22
	G	27.84	27.16	25.97	22.03	21.65	21.70	24	23	22	20	20	20
	H	30.57	30.30	30.16	29.39	28.64	28.38	26	27	26	25	24	24
		28.18	**27.82**	**27.03**	**24.05**	**23.73**	**23.85**	-	-	-	-	-	-

Table 3.3 – Summary statistics across α when $\beta=1$ for each construction heuristic

It is noted that the optimal solution for EAR (22 colours) was not obtained when $\beta=1$, but as β gets bigger the number of optimal solutions does not increase significantly. To gain greater insight into the performance of the algorithm across the construction heuristics, a sequence of barcharts have been constructed that display the proportion of feasible solutions of all timetables (HEC - 18 colours or less, EAR – 24 colours or less and TRENT – 23 colours or less) produced for each data set for each experimental value of α.

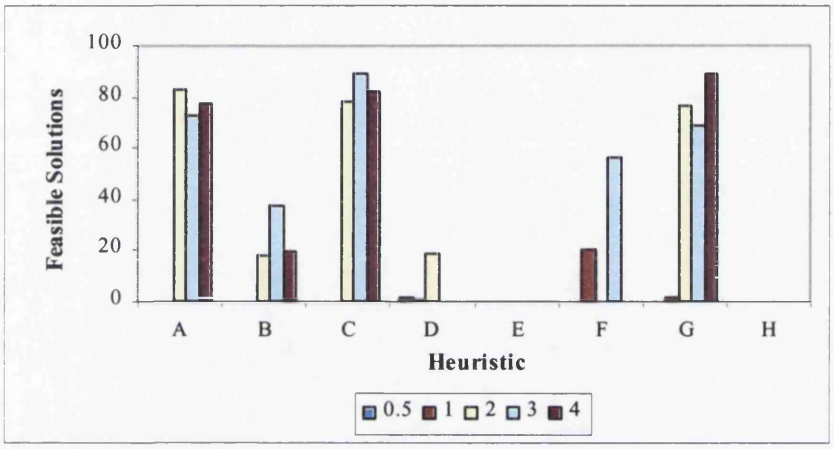


Figure 3.1 – Proportion of Feasible Solutions across α when $\beta = 1$ for HEC

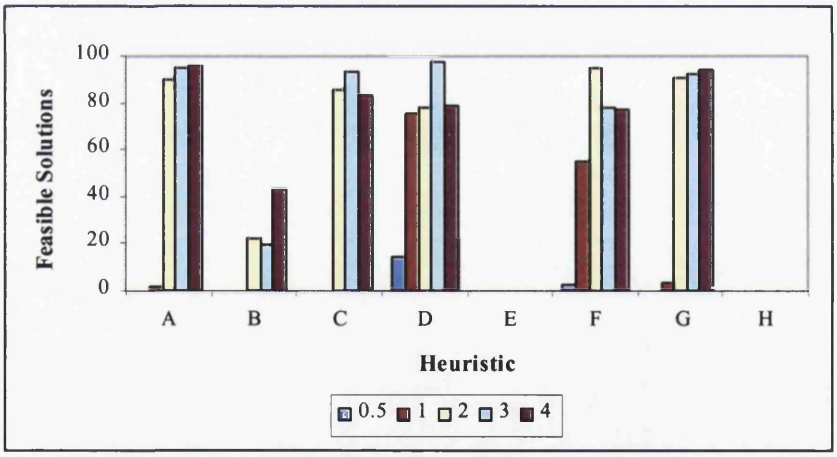


Figure 3.2 – Proportion of Feasible Solutions across α when $\beta = 1$ for EAR

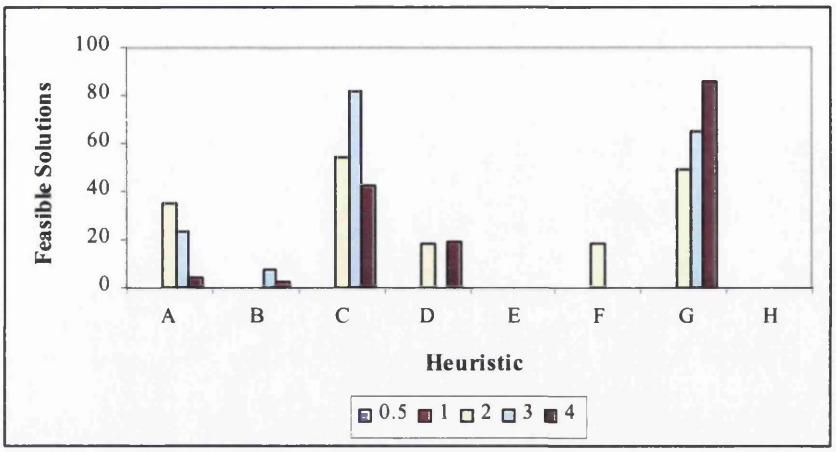


Figure 3.3 – Proportion of Feasible Solutions across α when β = 1 for TRENT

Figures 3.1-3.3 consider $\beta=1$ for a range of α, while the Figures relating to other settings of β can be viewed in Appendix 3.3 (Figures A3.1-A3.9). Firstly, heuristic *C* appears to be the forerunner of the construction heuristics (this is shown further in Appendix 3.1). Each barchart shows that using the ant algorithm while utilising construction heuristic *C* generates a high number of feasible solutions. We also note that the construction of solutions with the number of colours equal to the chromatic number is also possible through this heuristic. Heuristics *A*, *C* and *G* are the superior heuristics and outstrip *D*, *E* and *F* that were the leading single pass heuristics (when no trail was used). Heuristics *D-F* do produce some feasible solutions, but these construction heuristics encourage stagnation and this factor alone could have a detrimental effect on future investigations due to lack of ant exploration. It appears that randomising the selection of the first vertex allocated to a colour enhances the search process of the ants. Evidence supporting this can be found in Section 3.3. Heuristics *D* and *F* perform well for the EAR data set. Heuristic *E* struggles, while *H* returns the poorest solutions of all construction procedures.

Within each graph, each construction type has been broken down into values of α. As hinted before, the Figures suggest that $\alpha=2,3,4$ produce superior solutions but encourage results of similar quality and suggest that no great advantage is gained in increasing the measure of α. Increasing β does appear to enhance solution quality. Heightening β places greater influence on the desirability scores and compensates, to some extent, when low α is used but as α is increased the additional benefit on solution quality for higher β is minimal. Figures A3.10-A3.12 (Appendix 3.3) illustrate the *Average Colours* versus *Cycle* relationships for each of the data sets. These Figures depict the quicker improvement in

solution quality as α increases. Generally, achieving good quality solutions as early as possible is beneficial but this may encourage stagnation, which in the context of this chapter is desirable but when multiple objectives are considered stagnation becomes less favourable due to lack of exploration.

Figures 3.4-3.6 illustrate the average number of colours required to colour the graph over each cycle for each construction heuristic (with $\alpha=2$, $\beta=1$). Each Figure represents one data set (HEC, EAR and TRENT). The three Figures depict common trends. Solution quality improves as the search matures. Consequently, this indicates that the ants deposit feedback that is used positively by other ants later in the search. Generally, the average colours versus cycle graphs exhibit learning, sometimes intense, during the infancy of the search. The improvement in solution quality dampens as the search matures. This varies between the construction heuristics but it is safe to conclude that improvement stops relatively early in the full allocation of *100 cycles*. As stated previously, heuristics *D-F* choose the first vertex deterministically in each new colour and this contributes to the superior starting solutions. However, these heuristics are hindered by the lack of additional variation and the Figures show the early experience of stagnation while using these heuristics. Meanwhile, the heuristics with superior finishing solutions begin with relatively poor starting solutions.

Unless stated otherwise, $\alpha=2$, $\beta=1$ and construction heuristic *C* will be used in future studies. A comparison with the results published in Costa and Hertz (1997) can be found in Section 3.4.

3.2.3 Influence of Evaporation Factor

The evaporation factor ρ allows the user to gain a balance between exploration and exploitation. The factor ρ represents a rate of learning experienced by the ants. The higher the learning/retention rate the greater amount of trail that is made available to future ants as they construct solutions. Very small measures of ρ indicate that decisions are biased only to the recently generated solutions. The results tabulated in Table 3.4 represent varying ρ with $\alpha=2$ and $\beta=1$ under the construction environment of heuristic *C*. Further results can be obtained in Appendix 3.4.

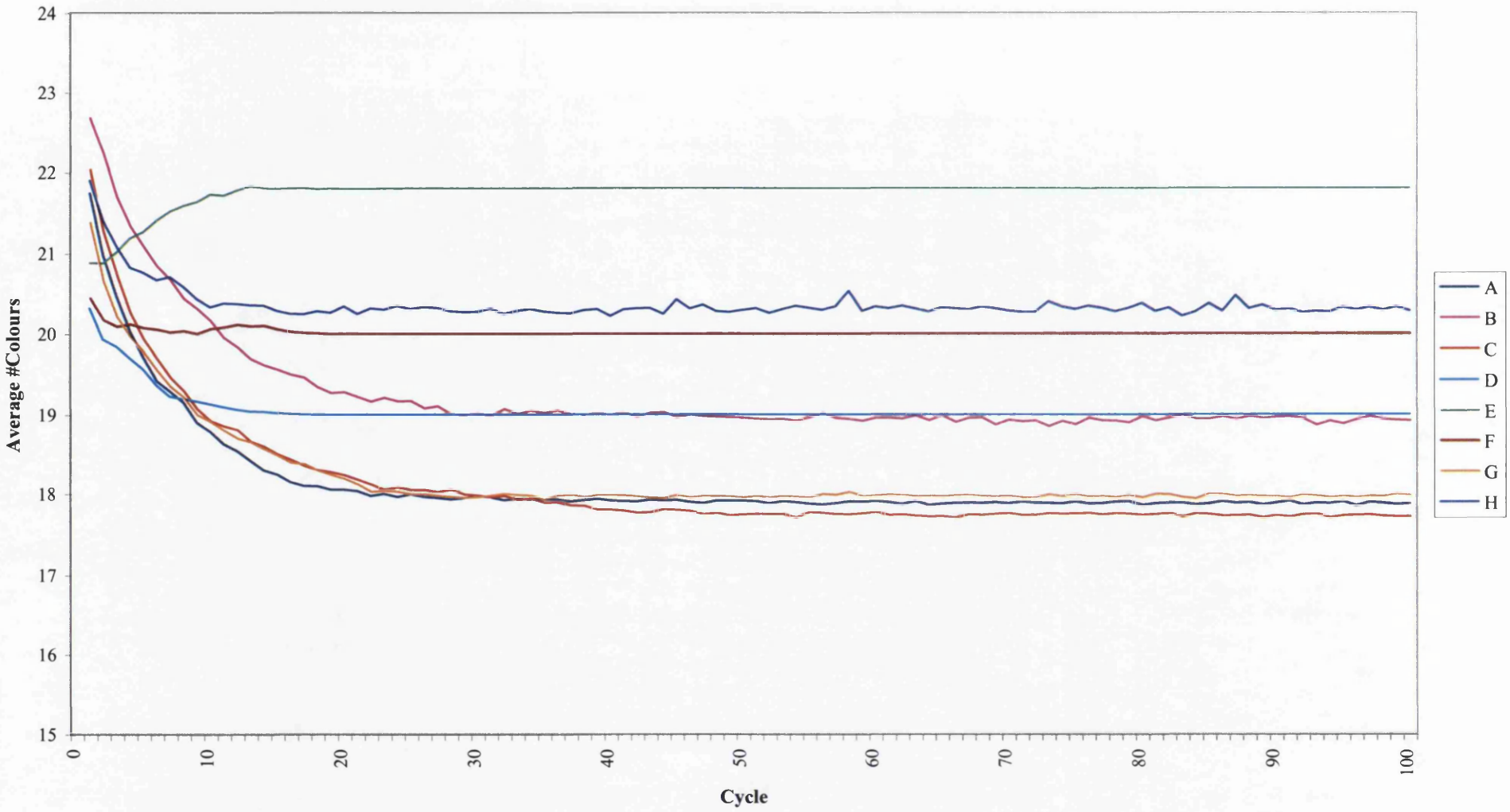


Figure 3.4 - Comparison of Construction Heuristics A-H for HEC

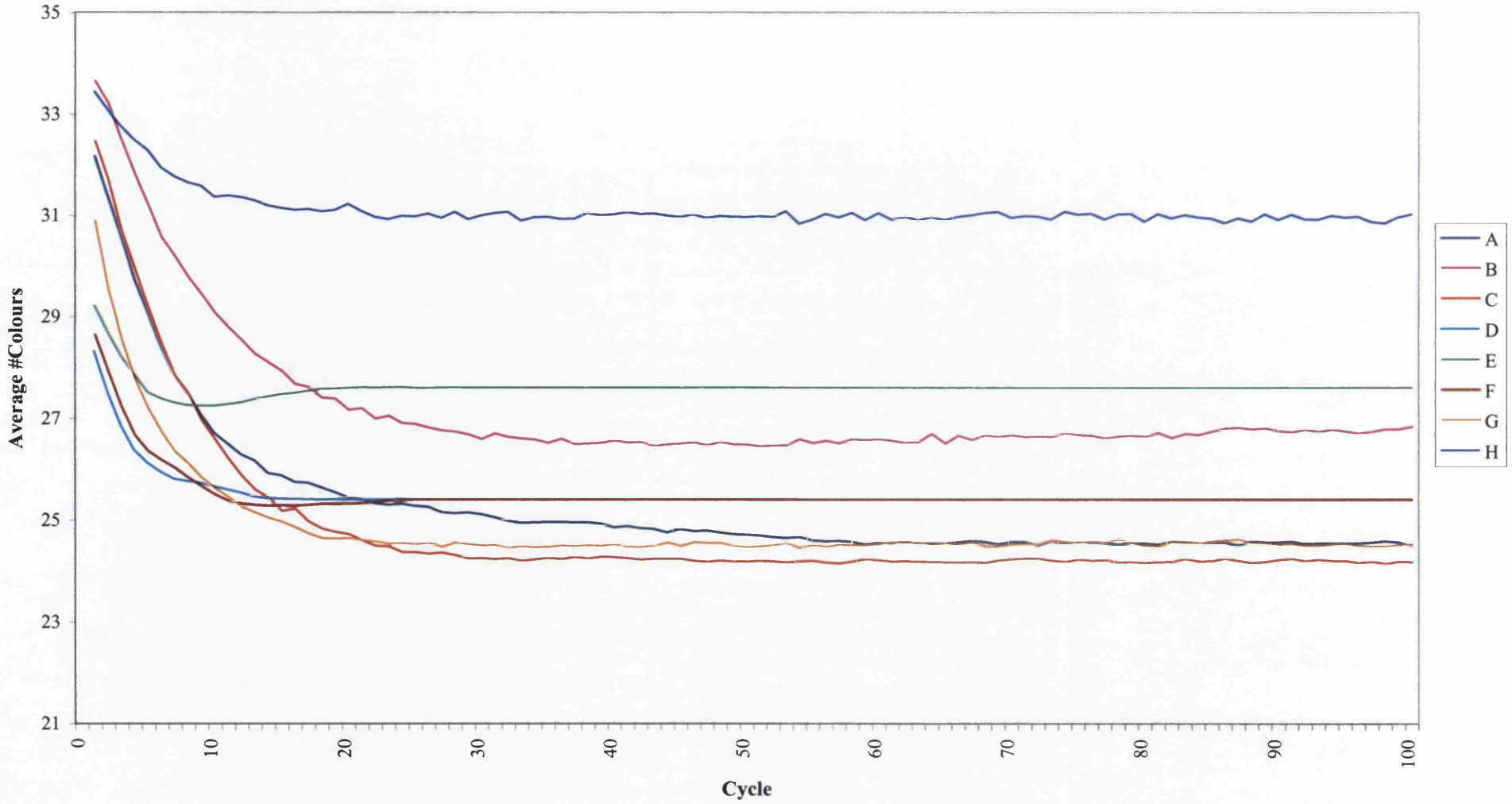


Figure 3.5 - Comparison of Construction Heuristics A-H for EAR

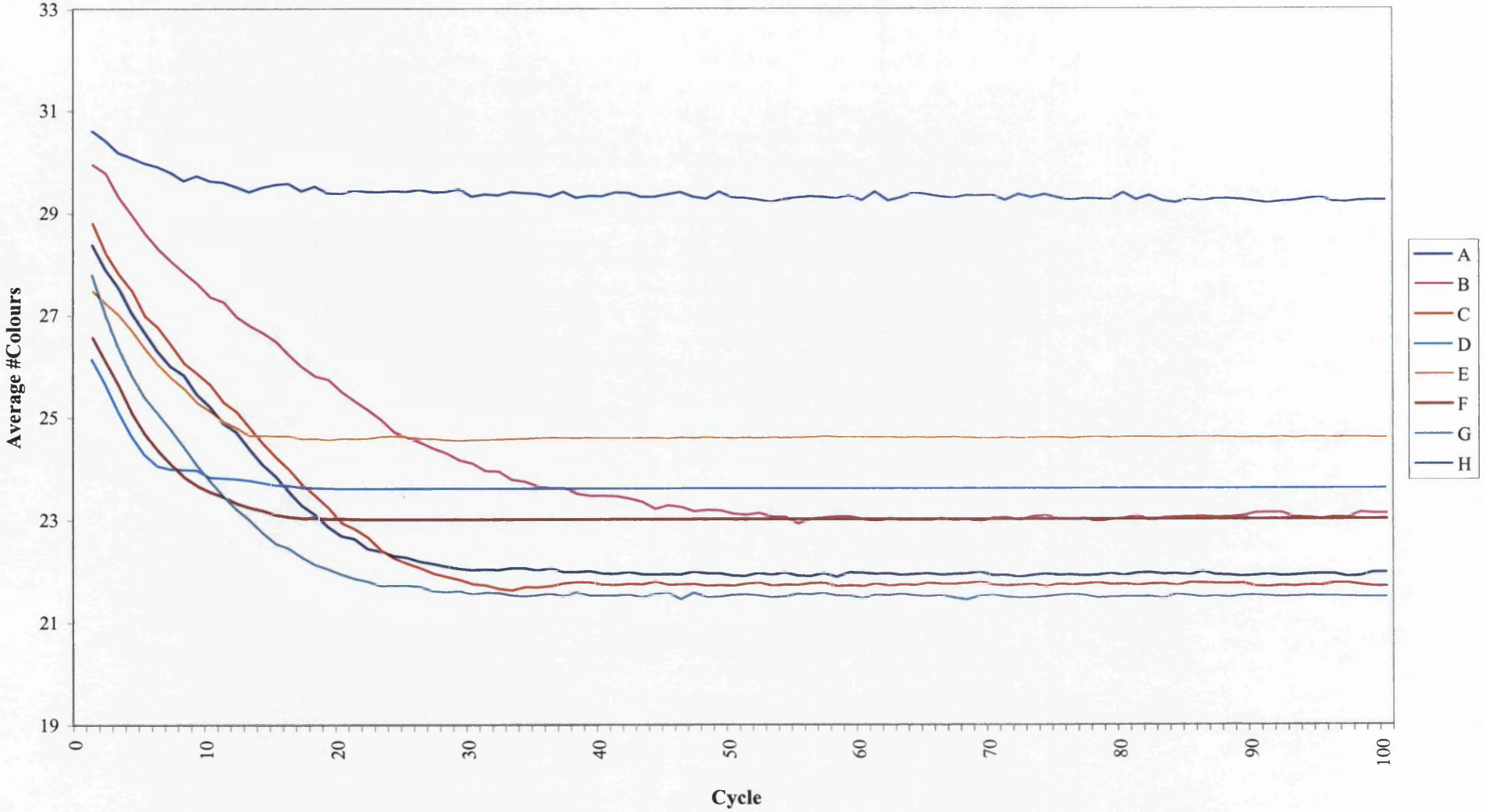


Figure 3.6 - Comparison of Construction Heuristics A-H for TRENT

	HEC		EAR		TRENT	
ρ	*Av*	*Best*	*Av*	*Best*	*Av*	*Best*
0.01	17.77	17	24.60	23	22.29	20
0.1	18.08	17	24.96	23	22.38	20
0.3	18.30	17	25.13	23	22.19	20
0.5	18.05	17	25.23	23	22.29	20
0.7	18.45	17	25.37	23	22.98	20
0.9	18.65	17	26.07	22	24.39	20
0.99	19.37	17	27.88	23	26.42	22

Table 3.4 – Summary statistics across ρ with construction heuristic C

Generally, $\rho \in [0.01, 0.7]$ is fairly robust since solution quality is consistent. Relatively good average solutions appear to be achievable with little memory. A common theme is the deterioration of average solution quality for higher ρ. Despite this, the optimum solution of 22 colours was obtained for the EAR data set when $\rho=0.9$. However, this was a 'one-off' observation and not a general occurrence. Costa and Hertz suggested that $\rho=0.5$ is suitable. This parameter setting may not produce the best solution but results are very robust within $[0.01, 0.7]$ and, consequently, it does seem counterproductive to argue for $\rho \neq 0.5$.

3.2.4 Efficiency of Method

In this section we will gain insight into the speed of convergence to the optimum or best-run solution during each run for each of HEC, EAR and TRENT. Table 3.5 shows the best-found solution and the cycle when it was first found. The results have been obtained under the run conditions of construction heuristic *C*, $N_A=N_C=100$, $\alpha=2$, $\beta=1$ and $\rho=0.5$.

	HEC		EAR		TRENT	
Run	*Best*	*First Found*	*Best*	*First Found*	*Best*	*First Found*
1	17	10^{th} cycle	23	14^{th} cycle	21	18^{th} cycle
2	17	72^{nd}	23	14^{th}	21	20^{th}
3	18	5^{th}	24	10^{th}	20	22^{nd}
4	17	8^{th}	23	13^{th}	21	20^{th}
5	17	8^{th}	23	9^{th}	20	20^{th}

Table 3.5 – Statistics describing the timing of discovery of run-best solution

With the exception of one experimental run (HEC – Run 2), the best run solutions were discovered early. For HEC, four experimental runs returned the optimal solution with three of these runs achieving this mark within ten cycles. For EAR, the algorithm failed to generate the known optimal solution of 22 colours. However, four runs managed to colour the graph in 23 colours, which is still lower than the timeslot threshold of 24 colours.

Again, these run-best solutions were found early on but the additional search time allowed did not result in improving these solutions. For TRENT, the optimal solution was observed during two of the runs, while the other three runs noted run-bests of one colour above the optimal (21 colours). The run-best solutions were generally found around the 20^{th} cycle mark. So, the bulk of the search was obsolete and consequently, computationally inefficient.

To conclude, the run-best is not always the optimal solution but advancements that are made (as detailed in Chapter 4) to the ant algorithm will prove that the optimal solution is achievable. This is desirable since better solution quality (at this stage) will potentially allow greater flexibility when considering additional and alternative objectives e.g. second-order conflict (see Chapter 5). It has been demonstrated that the run-bests were generally found during the infancy of the search and this suggests that N_C could be lowered. However, $N_C=100$ will still be used during future studies since solution method enhancements will require the use of extra runtime.

3.2.5 Ants per Cycle N_A

When solving the TSP, Dorigo et al. (1991) for example, suggested that the number of ants per iteration should be set near to the number of cities to achieve optimal results while Gambardella et al. (1999a) set the number of ants equal to the number of vehicles for the Vehicle Routing Problem. Such comments would suggest that, in general, the number of ants should be approximated near to the size of the problem, in our case the number of exams (vertices). The number of ants per cycle, N_A, influences both solution quality and the computational expense of the algorithm. The computational effort will vary linearly within each cycle according to the volume of ants chosen. In this section, we will detail the influence of N_A through Table 3.6 (Appendix 3.5 provides a more detailed tabulation of solution according to N_A and by construction heuristic).

	HEC (vertices=80)				EAR (vertices=189)				TRENT (vertices=261)			
N_A	*Av*	*Best*	*Opt Runs*	*Time (Secs)*	*Av*	*Best*	*Opt Runs*	*Time (Secs)*	*Av*	*Best*	*Opt Runs*	*Time (Secs)*
1	18.91	18	-	1.05	27.93	25	-	3.02	25.25	22	-	6.32
5	18.59	18	-	5.26	26.55	23	-	15.08	23.52	21	-	31.60
10	18.29	17	4	10.53	25.79	23	-	30.16	23.26	20	1	63.21
25	18.13	17	4	26.31	25.01	23	-	75.40	22.36	20	2	158.02
50	18.38	17	4	52.63	25.16	23	-	150.80	23.01	20	2	316.05
75	17.99	17	4	78.94	24.85	23	-	226.20	22.52	20	3	474.07
100	18.23	17	4	105.25	25.27	23	-	301.60	22.67	20	3	632.09
150	18.01	17	5	157.88	24.67	22	1	452.40	22.49	20	3	948.14
200	18.13	17	5	210.50	24.82	22	1	603.20	22.68	20	3	1264.18

Table 3.6 – Statistics describing the influence of the Number of Ants per Cycle

With the HEC and TRENT data sets, optimal solutions were discovered with relative ease. In both instances, setting $N_A=10$ achieves the desired results while a larger cycle population is needed for EAR to obtain the optimal solution of *22 colours*. The *Opt Runs* columns represent the number of independent runs that observed at least one optimal solution. This statistic is used to represent a measure of efficiency of the algorithm. If a scheduler wished to apply the ANTCOL algorithm to a problem instance then they may desire to perform only one run and the *Opt Runs* statistic indicates the likelihood of observing the optimal solution in any particular run. For all data sets, we note that as N_A increases *Opt Runs* also increases. However, the influence of N_A on average solution quality does dampen as N_A gets larger (as illustrated by the *Av* statistics). Despite this, observing that larger N_A does not have a detrimental effect on solution quality, it could be argued that a bigger N_A will increase the probability of obtaining at least one feasible timetable per run. With HEC, a greater number of runs observe the optimal and this can be associated to the size of the data set. More successful runs are observed with TRENT than EAR, but this can be attributed to the greater difficulty of the EAR data set as represented by the higher density. As stated earlier, Dorigo et al. (1991) suggested that putting the number of ants equal to the number of cities in the TSP is a sensible guideline. Similarly, Costa and Hertz (1997) used cycles of *100 ants* for graphs with *100 vertices*. However, Table 3.6 indicates that relatively good quality solutions can be achieved via N_A that is smaller than the size of the problem, thus reducing computational effort. However, it appears plausible that more difficult data sets will require extra ants per cycle to enhance the overall exploratory power of the ants. The difficult EAR data set is an example of this.

It is demonstrated that the relationship between N_A and runtime is linear. For example, $N_A=100$ requires twice the effort that $N_A=50$ does.

Unfortunately, there is no exact rule that can be applied to deducing N_A. The choice of this parameter depends on the size of the problem and the difficulty of solving the problem. If the scheduler wishes to perform a series of experiments then a little trial and error will point towards a sensible setting of N_A, which will allow for faster computation.

3.2.5.1 N_A-N_C Trade-Off

In this section, we note the benefit of cyclic trail updates on solution quality. For each $N_A \in [1, 5, 10, 25, 50, 75, 100, 150, 200]$ and each $N_C \in [10, 25, 50, 75, 100]$ the average and minimum number of colours needed to colour the HEC, EAR and TRENT graphs are presented. This cross tabulation will highlight two points, firstly, the importance of N_C on solution quality and secondly, if the overall number of ants is limited then how many cycles should be used to accommodate these ants.

		Av					Best				
	N_A/N_C	*10*	*25*	*50*	*75*	*100*	*10*	*25*	*50*	*75*	*100*
HEC	*1*	20.32	19.48	18.86	18.65	18.51	18	18	17	17	17
	5	20.38	19.50	19.02	18.84	18.74	18	18	18	18	17
	10	20.23	19.94	18.92	18.68	**18.52**	18	18	17	17	17
	25	20.18	19.25	18.75	18.54	18.45	18	17	17	17	17
	50	19.98	19.14	18.64	18.46	18.37	17	17	17	17	17
	75	19.77	18.91	18.52	18.38	18.31	17	17	17	17	17
	100	**19.80**	18.91	18.44	18.29	18.21	17	17	17	17	17
	150	19.98	19.22	18.75	18.61	18.54	17	17	17	17	17
	200	19.98	19.28	18.90	18.78	18.72	17	17	17	17	17
EAR	*1*	29.82	27.82	26.56	25.98	25.67	27	25	23	23	23
	5	28.50	26.36	25.30	24.97	24.79	24	23	23	23	23
	10	28.23	26.10	25.13	**24.81**	24.64	24	23	23	23	23
	25	27.79	25.93	25.13	24.85	24.72	24	23	23	23	23
	50	27.61	25.84	25.05	24.77	24.61	23	23	23	23	23
	75	**27.47**	25.80	24.90	24.55	24.37	24	23	23	23	23
	100	27.45	25.86	25.18	24.96	24.85	23	23	23	23	23
	150	27.44	25.84	25.19	24.95	24.82	23	23	23	23	23
	200	27.56	25.94	25.15	24.87	24.73	23	23	23	23	23
TRENT	*1*	26.96	25.31	24.32	23.89	23.61	24	22	21	20	20
	5	26.48	24.65	23.60	23.20	22.98	23	21	21	21	21
	10	26.12	24.25	22.95	22.42	22.11	22	21	20	20	20
	25	25.82	23.89	**22.72**	22.31	22.11	22	20	20	20	20
	50	25.64	**23.65**	22.47	22.06	21.84	21	20	20	20	20
	75	25.45	23.62	22.51	22.13	21.93	21	20	20	20	20
	100	25.45	23.56	22.55	22.20	22.03	21	20	20	20	20
	150	25.44	23.51	22.45	22.09	21.92	21	20	20	20	20
	200	25.57	23.73	22.43	21.97	21.73	22	20	20	20	20

Table 3.7 – Cross Tabulation of N_A against N_C

Table 3.7 indicates that higher N_C improves solution quality but as N_C increases the improvement dampens. This represents the early intense learning phase by the ants and then the inability to improve solution quality during the later stages of the search. These observations have been noted before.

If the number of ants is limited and distributed over different N_C, Table 3.7 indicates that higher N_C is preferable. Let us consider three examples, which are bolded. Firstly, for HEC, if the total number of ants is levied at *1000 ants*, it is noted that allocating *100 ants* across *10 cycles* returns an average solution of *19.80 colours* while, placing *10 ants* over *100 cycles* needs an average of *18.52 colours*. Secondly, for EAR, if the number of ants is set at *750*, allocating *75 ants in 10 cycles* requires an average of *27.47 colours* while the reverse, produces an average of *24.81 colours*. Finally, for TRENT, setting N_A to *50* and N_C to *25* needs an average *23.65 colours* while setting N_A to *25* and N_C to *50* gives an improved average of *22.72 colours*. These examples represent a common trend and highlight the importance of trail replenishment updates, which when performed frequently enough directs a more focussed search process and encourages better solution quality. Greater cyclic trail replenishment updates will incur greater computational effort but the increase is not significant and should not deter a user from increasing N_C if the total number of ants is fixed. It is concluded here that for a fixed number of ants, it is more efficient to allow for a greater number of trail updates i.e. allow for more N_C. However, limitations to this rule of thumb do exist.

3.3 Stochastic Importance

At a vertex-colour decision point, each feasible vertex is labelled with a probability that represents the potential benefit of inserting a vertex into the solution at that stage. These probabilities are generated through the RPR and decisions are made according to the well-known 'roulette wheel' philosophy, see Goldberg (1989) for discussion. Thus, ant algorithms, to some extent, hold a stochastic basis and do not simply select deterministically the 'perceived best' vertex insertion at a vertex-colour decision point. This deterministic score would represent the largest combined score of trail and desirability (biased to powers α and β respectively). In this section, we note the improved solution quality that a stochastic approach offers, which is illustrated in the Figures 3.7-3.9.

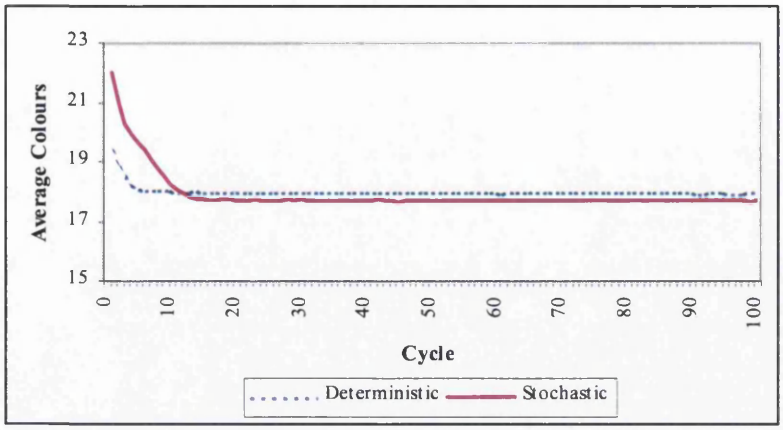


Figure 3.7 – Deterministic against stochastic comparison for HEC

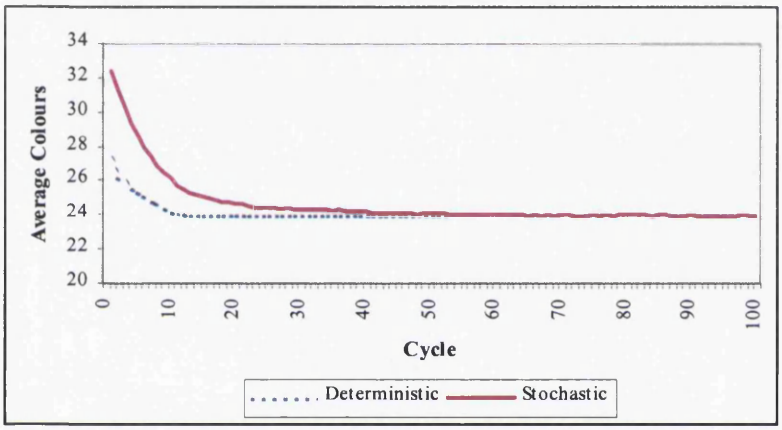


Figure 3.8 – Deterministic against stochastic comparison for EAR

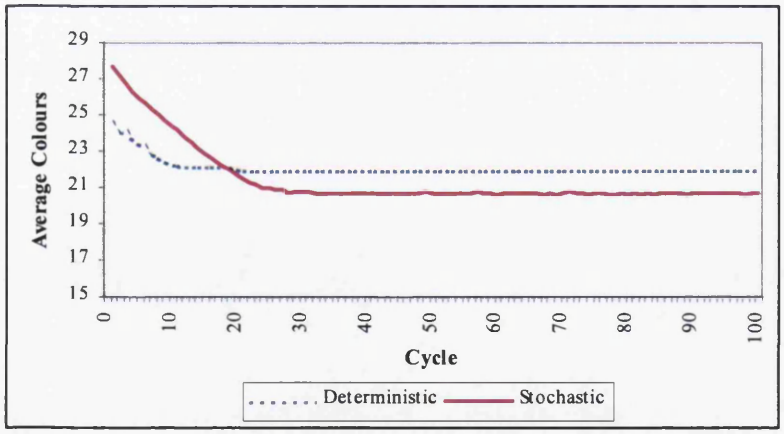


Figure 3.9 – Deterministic against stochastic comparison for TRENT

Figures 3.7-3.9 depict similar traits. Both procedures show evidence of learning during the infancy of the searches. Improvement dampens relatively early in the explorations. The deterministic approach generates superior starting solutions, which is to be expected since, as the first cycle in any search does not utilise any trail information, all constructions are

made according to the desirability factor alone. A deterministic method will intuitively create better solutions at the start of the search due to the more controlled environment it offers i.e. biasing selection towards desirability scores. Randomisation encourages wider exploration and explains the better final solutions that are constructed. The stochastic method does allow for a greater combination of constructed timetables and encourages the avoidance of stagnation that is commonplace with deterministic approaches. Randomisation plays a pivotal role in the success of method since it ensures wider exploration, which is proven to be key when observing additional objectives such as second-order conflict (Chapter 5).

3.4 Comparison with Costa and Hertz (1997)

Since the ANTCOL variant (first) presented in Costa and Hertz (1997) forms the basis of not only the work in this chapter but the entire thesis, it is sensible to draw comparisons with the results stated in the above mentioned paper. Initial experiments were performed on 20 graphs of 100 vertices with a density of *50%*. The authors showed that setting $\alpha=2$ and $2\leq\beta\leq4$ for RLF based tests derived superior results. However, the impact of β was not too dissimilar to that witnessed through experiments in this chapter, although it has been suggested here that $\beta=1$ is more suitable due to the advantage of reduced runtimes. Meanwhile, it was claimed in Costa and Hertz (1997) that $\alpha=1$ was more favourable for applications of DSatur. The authors also noted that better solutions were obtainable for RLF heuristics *D*, *E* and *F* than *A*, *B* and *C* for lower α but the reverse applies for higher α. Results suggest that the effectiveness of the construction heuristics is similar to what has been experienced here. Construction heuristic *C* is again the forerunner but there is a greater similarity between *A* and *B* than witnessed in our trials. With respect to DSatur, *G* outperforms *H* with *G* again competing with the better RLF heuristics (those that insert first vertex in a colour on a random basis). It was also demonstrated that setting $\rho=0.5$ is the more appropriate setting but it was observed that this parameter does not influence the performance of ANTCOL, which corresponds to our observations. The authors noted that solution quality improves as the number of ants per cycle approaches the size of the graph. However, if N_A goes above the size of the graph, the effect on solution quality is minimal, but not detrimental. But, the authors did quote that large N_A could lead to slower search convergence when applying DSatur based construction heuristics. During our experiments, this has been noted but on a smaller scale. Finally, similar conclusions have been drawn

regarding the importance of randomness during construction. The authors demonstrated that significantly poorer results are achieved through deterministic selection of vertices at each vertex-colour decision stage. In summary, many of the conclusions reached here correspond to those presented in Costa and Hertz (1997).

It should be noted here that the graphs used within this thesis are different in nature to those presented in Costa and Hertz (1997) and therefore, may lead to any dissimilar conclusions that have been presented. Costa and Hertz used random graphs whereas the graphs used here are of examination timetabling origin and will undoubtedly contain larger maximal cliques, thus increasing difficulty, Carter et al. (1994).

3.5 Mixing Construction Heuristics

From previous investigations, it was proven that the capabilities (when used in conjunction with trail) of the eight construction heuristics were similar when compared across the data sets. Construction heuristics *A*, *C* and *G* were consistently the better achievers across the data sets, while *E* and *H* were consistently poor. However, it has been demonstrated that heuristics *D*, *E* and *F* are the superior single pass heuristics (see Appendix 3.1). The reasoning is associated with the superior starting solutions that these heuristics generate. This section attempts to take advantage of this characteristic. During each cycle, two pre-selected heuristics are made available for use. If there are N_A ants per cycle and if n ants use one construction heuristic during a cycle then $(N_A\text{-}n)$ ants use an alternative heuristic.

Heuristic *C* has been observed the forerunner when used with trail and will initially be used along with the other superior heuristics *A* and *G*. We will find that these amalgams do not offer any significant extra constructive power for EAR and TRENT and consequently, heuristic *C* is used along with the more efficient single pass construction heuristics, *D* and *F*. The *FG* amalgam was also attempted due to the promising characteristics that these heuristics exhibit when used independently with TRENT. However, these results will not be presented here, as the results were not superior to those experienced with *CG*.

Each cycle is divided into two parts with C_A ants utilising heuristic *C*, while A_A, G_A, D_A or F_A ants using heuristics *A*, *G*, *D* or *F* respectively. The amount of ants allocated to each

heuristic type is varied and we will observe the sensitivity in solution quality as a result of this variation.

			Construction Heuristic Amalgams							
			CA		*CD*		*CF*		*CG*	
	C_A	-	**Av**	**Best**	**Av**	**Best**	**Av**	**Best**	**Av**	**Best**
HEC	*0*	*100*	18.14	17	19.06	17	20.02	18	18.21	17
	10	*90*	18.21	17	19.24	17	18.88	17	18.38	17
	30	*70*	18.21	17	18.49	17	18.65	17	18.29	17
	50	*50*	17.99	17	18.54	17	18.38	17	18.31	17
	70	*30*	18.32	17	18.25	17	18.21	17	18.30	17
	90	*10*	18.15	17	**17.97**	**17**	18.00	17	18.16	17
	100	*0*	18.09	17	18.09	17	18.09	17	18.09	17
EAR	*0*	*100*	25.27	23	25.50	24	25.51	24	24.85	23
	10	*90*	25.21	23	25.30	24	25.17	23	24.84	23
	30	*70*	24.80	23	24.28	23	25.13	24	24.62	23
	50	*50*	25.11	23	24.76	23	24.35	23	24.81	23
	70	*30*	25.08	23	24.25	22	24.04	22	24.79	22
	90	*10*	24.85	23	**23.94**	**22**	24.10	22	25.02	23
	100	*0*	24.81	23	24.81	23	24.81	23	24.81	23
TRENT	*0*	*100*	22.62	20	23.70	22	23.19	22	22.03	20
	10	*90*	22.52	20	22.98	21	22.67	21	21.89	20
	30	*70*	22.55	20	22.67	21	23.03	21	21.99	20
	50	*50*	22.52	20	22.58	21	22.16	21	**21.89**	**20**
	70	*30*	22.49	20	22.00	20	22.25	20	22.30	20
	90	*10*	22.64	20	22.32	20	22.33	20	22.53	20
	100	*0*	22.53	20	22.53	20	22.53	20	22.53	20

Table 3.8 – Statistics describing efficiency of selected construction heuristic amalgams

The above set of results show that there is some benefit from using two construction heuristics, however any improvements are minor. In particular, construction heuristic amalgams CD and CF offer additional solution building power when C_A=*70* and *90* and notably, the optimal solution of *22 colours* is observed with the EAR data set. The amalgam CG generates better solution quality for TRENT. The best result for each set is bolded. Figures 3.10-3.12 illustrate these best results for the average versus cycle relationship for each data set.

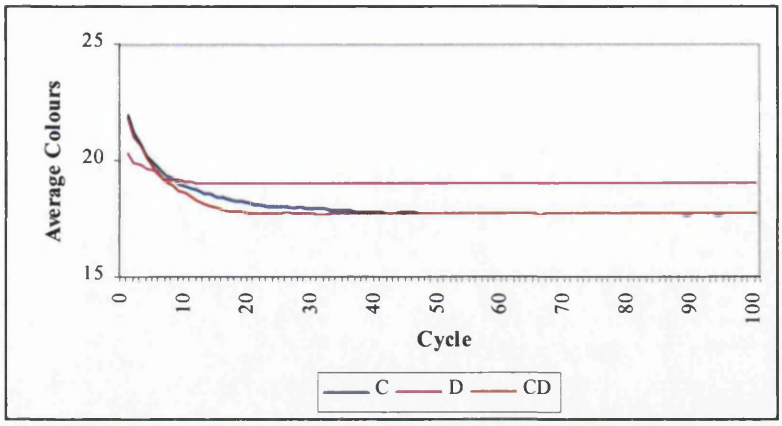


Figure 3.10 – Average solution quality for heuristics C, D and amalgam CD for HEC

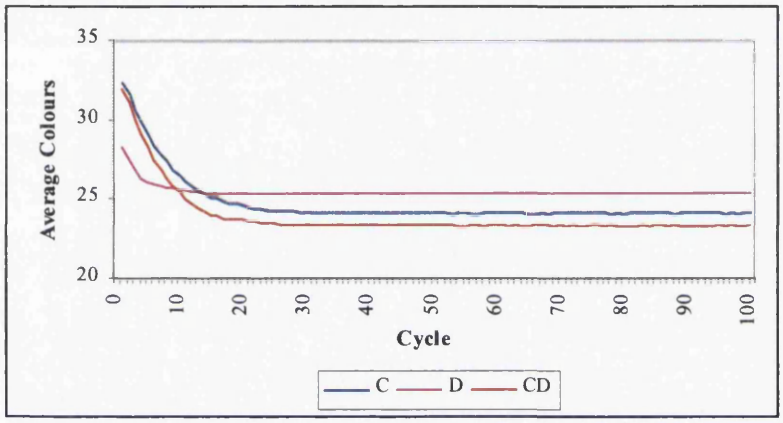


Figure 3.11 – Average solution quality for heuristics C, D and amalgam CD for EAR

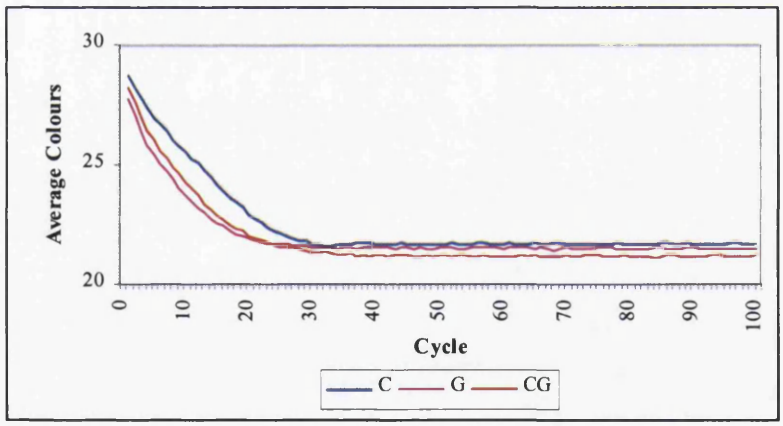


Figure 3.12 – Average solution quality with heuristics C, G and amalgam CG for TRENT

The lines labelled CD in Figures 3.10 and 3.11 indicate that more focused learning occurs during the infancy of the search process and this can be associated with the involvement of the more greedy Heuristic D. Due to the majority of the ants utilising Heuristic C the superior constructive power still remains and is enhanced due to the more deterministic nature of Heuristic D. With regards to overall final average solution, the CD amalgam does not enhance solution quality with HEC but an improvement is certainly notable with the

EAR data set, which suggests that colouring harder graphs may benefit from such a heuristic collaboration. With respect to TRENT, it was discovered previously that the best heuristic was G and the superior RLF heuristic that deterministically selects the first vertex in each colour group was F. Based on the success of the CD amalgam one would expect similar success when using FG for TRENT. However, the results indicated otherwise and the best results belonged to the CG amalgam. To generalise this approach, it seems wise to suggest the use of the CD amalgam due to its consistency over all data sets. The use of three heuristics within cycles has been attempted but did not offer any additional constructive power.

3.6 Trail Potency

Here we examine the extent that trail levels influence the selection procedure of ants during the search. A measure of quality (Equation 3.2) was introduced in order to quantify the impact of good solutions and the effect that good or poor solutions have upon the direction of the search. Such a measure was considered for each pair of vertices v_r and v_s over each cycle. Given m ants per cycle, the colourings generated within that time are $s_1,\ldots,s_m$ and $S_{rs} \subseteq \{s_1,\ldots,s_m\}$ denotes the subset of solutions in which v_r and v_s were allocated to the same colour group. Let q_t be the number of colours used in s_t ($1 \leq t \leq m$) and *worst* represents the solution of poorest quality observed during an experimental run. A random non-adjacent pairwise set (v_r, v_s) from the HEC data set is used to illustrate the relationship between the two factors of interest.

$$Quality_{rs} = \sum_{s_t \in S_{rs}} (worst - q_t) \quad (3.2)$$

Each score (*Quality*) represents the sum of the difference between the worst observed colouring and each observed colouring returned by individual ants – the higher the measure, the more potent the relationship.

Analysis is performed that examines the impact of quality for a pair of vertices in the n^{th} cycle and the response in the number of ants that allocate the same colour to that pair of vertices during the $n+1^{th}$ cycle. It was felt that a good relationship during one cycle should bias the ants during the following cycle.

Figure 3.13 demonstrates the quality of a pair of vertices in the n^{th} cycle does determine, to a certain extent, the volume of ants that consider the same pair of vertices in the $n+1^{th}$ cycle. The positive relationship suggests that an increase in quality during the n^{th} cycle encourages more ants to include the same pair of vertices within the same colour group. The relationship appears to be fairly linear and does not exhibit a step function appearance that would indicate that different rates of impact upon the number of ants across intervals of quality.

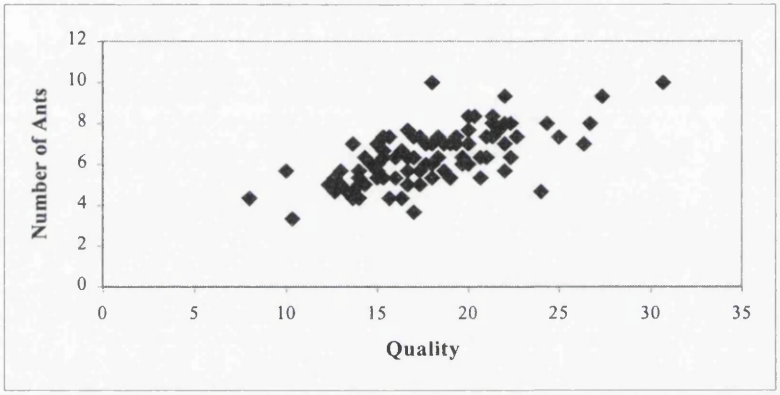


Figure 3.13 – Number of Ants versus Quality

The correlation coefficient associated with the information displayed in Figure 3.13 was calculated as 0.65, which is significant at the 5% level. This confirms that a positive linear relationship exists between quality in the n^{th} cycle and the number of ants selecting a pairwise set in the $(n+1)^{th}$ cycle and importantly indicates that the trail is biasing the decisions of the ants.

3.7 Speed Related Issues

This section is inspired by the computational expense that is associated with the ant algorithm, which immediately reduces the attractiveness of the procedure. Here, we attempt to speed up the process, which will enable the completion of more experiments and will make a real-life examination timetabling solution method more attractive to users. The greatest computational time consumer is the calculation of exponents and hence, great focus is placed on this.

The manner in which solution quality is represented now changes. Given that examination timetabling deals with *feasible* timetables, it was felt justified to record the volume of

feasible solutions achieved rather than *average* number of colours. A *feasible* solution (timetable) is one that allocates all the vertices (exams) within a maximum number of colours (timeslots) without violating any hard constraints (e.g. direct conflict). Within this section, the terms *Feasible* and *Time* will frequently be used and can be defined as follows.

Feasible – Average number of feasible timetables across five experimental runs.
Time – Average runtime across five experimental runs.

3.7.1 Revising the RPR

Experiments are only performed on the trail factor component in the RPR with the desirability factor acting as a constant i.e. raised to the power of 1. Firstly, it is proven that representing the value x^{α} as a complete multiplicative term $x.x.x..x$ reduces computational effort with no sacrifice to solution quality. The runtimes (in seconds) are presented below for both the full multiplicative and exponent based RPR expressions when $\alpha=2$ and $\alpha=3$.

	α=2		α=3	
	Multiplicative	*Exponent*	*Multiplicative*	*Exponent*
HEC	90.13	105.25	175.49	223.82
EAR	301.60	413.75	504.25	757.94
TRENT	475.75	632.09	898.23	1291.80

Table 3.9 – Runtimes of multiplicative and exponent based RPR expressions

It has been shown that significant computational effort can be saved by expressing terms multiplicatively rather than in exponent form.

Additionally, expensive computational effort is required to accommodate the decimal places associated with the trail scores and consequently, the effort required to process each cycle magnifies as the cycle count increases. In the early stages of a run, allowing the trail factors to be defined to a high degree of accuracy is critical due to the small differences in trail levels. Therefore, losing accuracy at this stage will have a detrimental influence on the overall success of the run and the algorithm reduces to a basic single pass heuristic. However, as the run matures the trail levels between options become wider and some accuracy can be lost without influencing solution quality greatly. Here, we investigate the benefit of reducing the degree of accuracy associated with the trail factor scores. An initial 'run-in' period will be allowed without accuracy manipulation and then the trail factors will be reduced to integers. The RPR is represented on a multiplicative basis due to the

above discussion. With HEC, runtime is reduced to 44.50 seconds with no decline in solution quality. Runs with the TRENT data set do not suffer on a quality basis and the required runtime reduces from 475.75 to 366.52 seconds. However, solution quality is lost with the harder EAR data set. The proportion of feasible solutions drops from 76.22 to 32.13 timetables per cycle but, positively, the average runtime reduce from 301.60 to 235.14 seconds.

This section illustrates what basic modifications can made be to the raw program set-up in order to reduce runtimes, but these tests have only been performed on Fortran 95 and assumptions cannot be made to alternative programming languages. In conclusion, it is demonstrated that using multiplicative terms rather than power exponents reduces computational effort. Additionally, it appears that simplifying the RPR that uses integer trail values is not suitable for the harder data set - EAR.

3.7.2 Linear Interpolation

Linear interpolation (*LI*) is a method that can be used for predicting an unknown value if you know any two particular values and assume the rate of change between these values is constant. *LI* will be used to estimate the trail component (scaled by the number of coloured vertices in colour under construction) of the RPR when raised to the bias parameter α. This trail component will be denoted by x. Continuous score x lies in the interval $[0, x_{max}]$ with x_{max} determined through preliminary investigation. This range is divided into sub-intervals i labelled by $int_i[x]$ with start point x_s and endpoint x_e. These interval extreme points are applied to $f(x)$, with $f(x)=x^{\alpha}$ and $\forall x \in [0,x_{max}]$ and stored. The interpolated $f(x)$ for x between the extremities is calculated as follows:

$$f(x) = f(\text{int}_i[x_s]) + \{f(\text{int}_i[x_e]) - f(\text{int}_i[x_s])\} * (x - \text{int}_i[x_s]) \quad (3.3)$$

After some preliminary investigation, it was decided that x_s and x_e will be fixed one unit apart. Reducing the length of interval $[x_s, x_e]$ requires inefficient runtimes, while increasing the length sacrifices solution quality. The values of $f(x)$ are read-in from a data file and act as a look-up table and are extracted when required and $f(x)$ is then interpolated accordingly. To illustrate the benefit of this approach, a range of values of α were

compared and the results are presented in Table 3.10. These results are compared directly against runs of the algorithm where no time saving implements are incorporated.

		Time		Feasible	
	α	*Full*	*LI*	*Full*	*LI*
HEC	*2*	105.25	65.10	83.58	84.91
	3	223.82	62.34	93.68	71.15
EAR	*2*	413.75	339.64	81.32	78.91
	3	757.94	344.28	85.24	88.52
TRENT	*2*	632.09	542.42	86.59	80.83
	3	1291.80	544.15	94.42	91.70

Table 3.10 – Feasibility and runtimes for Full and LI methods

The reduction in runtimes is significant with a very encouraging trade-off with solution quality i.e. the feasible timetables for *Full* and *LI* are not significantly different. The *LI* method appears to be attractive with the only drawback being the determination of the range of x values, which can be obtained with minimal effort. The use of *LI* for higher α should be treated with caution given that interpolation is performed within bigger intervals of $f(x)$. Typically, when $\alpha=2$ and $x=1.5$ we are required to interpolate between $f(1)=1$ and $f(2)=4$, however when $\alpha=5$ and $x=1.5$ it is necessary to interpolate between $f(1)=1$ and $f(2)=32$. Therefore, a greater scope for inaccuracies exists.

3.7.3 Candidate Lists

Dorigo et al. (1991) suggested the use of a candidate list structure when the Ant-Cycle algorithm was applied to the TSP. When at city x_t, the choice of the next city x_{t+1} to visit is restricted to a group of n cities, which belong to the candidate list. These n cities have the smallest desirability scores i.e. the smallest distance between city x_t and potential city x_{t+1}. A candidate list implementation is also used in conjunction with the Tabu Search (TS) meta-heuristic, Glover (1989). This tabu list is used to prevent repeating a recently constructed solution. This approach attempts to avoid stagnation and consequently, encourage search diversity.

Here, a candidate list structure is used to reduce the number of probabilities that are generated at a vertex-colour decision point. Each probability is computed through the RPR, which has components (trail and desirability) that are raised to bias powers, and the use of power exponents adds to the computational expense of Ant Algorithms. Since all vertex-colour assignments, except for the allocation of the first vertex in each colour group, makes

reference to the RPR, the additional computational expense is large and consequently, introducing means to reduce the use of RPR seems appropriate.

A series of lists with different attributes have been employed and we assess the trade-off between solution quality and computational saving. Candidate list membership will depend on the following attributes:

- Trail – at each vertex-colour decision point the vertices with the higher trail factor scores are selected.
- Desirability – at each vertex-colour decision point the vertices with the higher desirability factor scores are selected.
- Random – at each vertex-colour decision point no preference to any bias is given.

Lists of varying length and attribute combinations are investigated and the results are presented below.

	HEC		EAR		TRENT	
Candidate List	*Feasible*	*Time*	*Feasible*	*Time*	*Feasible*	*Time*
No List	83.60	105.25	76.22	413.75	83.61	632.09
10 Trail	70.14	82.59	32.94	324.57	87.73	542.84
20 Trail	87.24	97.60	35.70	365.89	87.46	574.44
30 Trail	86.95	100.35	32.03	360.07	90.14	603.33
10 Desirability	88.98	70.40	88.26	349.49	96.02	516.94
20 Desirability	91.71	120.72	87.74	432.33	93.55	632.70
30 Desirability	89.09	137.51	86.85	446.86	93.20	679.44
10 Trail/10 Desirability	69.83	108.56	84.77	427.32	92.32	647.45
10 Tra/10 Vis/10 Ran	89.17	121.87	84.03	455.03	91.77	630.47
10 Random	39.22	48.24	0.57	263.11	10.16	458.21
20 Random	58.23	53.72	55.09	290.46	69.67	495.38
30 Random	70.13	63.19	68.03	321.58	82.61	542.86

Table 3.11 –Feasibility and Runtimes for Full and Candidate list constitutions

Some candidate list constitutions are computationally more efficient. Candidate lists with members included only due to trail score are consistently faster than running the algorithm without a candidate list. Trail based constitutions generate very good solution quality for HEC and TRENT data sets but these constitutions struggle for the EAR data set. Candidate lists made purely of desirability-biased members improve solution quality with a list size of 10 reducing runtimes. The amalgam constitutions [*10 Trail/10 Desirability, 10 Trail/10 Desirability/10 Random*] are more than satisfactory with regards to solution quality but the

additional sorting that is required to identify members encourages runtimes greater than the *no list* scenario. Lists consisting of *Random* entrants reduce computational effort drastically but results do not compare with criteria-based list membership. As an example of the effectiveness of criteria-based candidate lists let us compare two list types – *10 Desirability* and *30 Random*. With HEC, *10 Desirability* generates an average of 88.98 feasible solutions in 70.40 seconds while *30 Random* needs an average run completion of 63.19 seconds and produces 70.13 timetables per 100 ants. With EAR, *10 Desirability* constructs 88.26 feasible solutions (over 12 more than *no list*) in a time of 349.49 seconds, while *30 Random* requires a slightly less time of 321.58 seconds to produce an inferior 68.03 feasible solutions. The most striking result is associated with TRENT. *10 Desirability* produces a significant 96.02 feasible solutions in an efficient 516.94 seconds while, *30 Random* requires more runtime (542.86 seconds) to generate a lower number of feasible solutions (82.61).

In conclusion, it can be seen that criteria-based candidate lists often encourage better solutions and, in some cases, require less computational effort to do so. From the candidate list constitutions evaluated, it seems that limiting ants at each decision vertex-colour allocation point to *10 Desirability*-biased vertices, if 10 feasible vertices are available, is the superior option.

A hybrid time conservation method has also been investigated. *LI* (when α=2) along with the preferred candidate list with respect to quality-time trade-off, *10 Desirability*, are used in one approach. Table 3.12 presents *Time* and *Feasible* information comparing the *hybrid* and *candidate list* methods.

	Time		**Feasible**	
	Hybrid	*10 Desirability*	*Hybrid*	*10 Desirability*
HEC	53.38	70.40	86.93	88.98
EAR	298.78	349.49	88.95	88.26
TRENT	483.70	516.94	95.86	96.02

Table 3.12 – Time and Feasible statistics for Hybrid and 10 Desirability time conservation methods.

The results in Table 3.12 indicate that additional time efficiency can derive from estimating the trail factor scores of *10 Desirability* vertices via *LI*. In addition, the differences between the *Feasible* rates between methods are insignificant, thus emphasizing the suitability of this application.

3.7.4 Limiting Trail Levels

High trail concentrates reduce the exploratory abilities of the ants due to the obvious bias towards certain components. Thus, small sections of the solution space may be investigated. This is commonplace as the search matures. Stützle and Hoos (1997) attempted to overcome limited ant foraging by stipulating maximum trail levels (τ_{max}) in the MMAS for the TSP. The authors also imposed a minimum trail level (τ_{min}) in order to encourage the selection of less popular edges within constructions, thus allowing for a wider search of the solution space. While it is always beneficial to produce diverse solution types, the use of a maximum trail level will be used purely for the purpose of reducing computational effort here. The storage of large trail scores is expensive and placing an upper bound on pheromone levels will have an effect of improving runtimes. However, we are also concerned in the trade-off with solution quality. Experiments have been performed for a small range of τ_{max} to gain an insight of the influence. The same τ_{max} have been used for each of the data sets as trail levels are fairly standard across data sets (see Figure 5.28 for trail levels).

	τ_{max}							
	No Limit		**10**		**50**		**100**	
	Time	*Feasible*	*Time*	*Feasible*	*Time*	*Feasible*	*Time*	*Feasible*
HEC	105.25	83.60	45.68	1.17	70.83	47.61	77.67	77.93
EAR	413.75	76.22	244.17	0.58	282.02	40.93	266.27	61.73
TRENT	632.09	83.61	475.05	2.79	478.63	66.68	455.27	83.07

Table 3.13 – Feasibility and Runtimes for full and various τ_{max}

The results indicate that the handling of larger trail levels slows the run speed of the algorithm. The aim is to impose a limit τ_{max} that quickens runtimes but does not significantly reduce solution quality. It is observed that setting τ_{max} *=100* allows for a very good trade-off between the factors of interest. For example, with TRENT, imposing a setting of τ_{max}*=100* reduces the runtime from 632.09 (with *no limit*) to 455.27 seconds while maintaining very similar feasibility levels. However, lower τ_{max} returns relatively poor solution quality. Typically, τ_{max}*=10* could not be used due to low feasibility despite relatively low runtimes. To gain perspective of the levels of highest trail, refer to Figure 5.28.

3.8 Conclusion

The aim of this chapter was to examine the influence of various components of the ANTCOL variant and to establish an appropriate environment as the basis for future experimentation. Studies within this chapter provide insight of the capabilities of the algorithm and the appropriate conditions that lead to superior search processes. It has been demonstrated that the trail bias parameter α has a significant influence on the capabilities of the algorithm. The desirability bias parameter β is pivotal for lower α but becomes less significant as α increases. The other main parameter of the ANTCOL variant, ρ (which dissipates trail levels proportionately), has little effect on solution quality. Another great determinant on search potency is the type of construction heuristic that is used. It is demonstrated that heuristic *C* is the most suitable due to its superior performance when considering all test data sets. Additionally, some extra constructive power can derive from mixing construction heuristics within a cycle. It is shown that using heuristics *C* and *D* within cycles generates the best results and superior performance can be attributed to the exploitation of the deterministic nature of *D*. However, it is demonstrated that limited use of this heuristic enhances solution quality the greatest. Under arguably the most appropriate settings $\alpha=2$, $\beta=1$ and $\rho=0.5$ and construction *C* (or *Amalgam CD*), **optimal or near-optimal solutions can be achieved at relatively early stages** in the search. Additionally, the *Best* solutions achieved are always better than the stipulated number of timeslots, which allows for greater flexibility when additional objectives are introduced, such as second-order conflict (Chapter 5).

A further study was performed to examine the effects of exploitation and exploration. Complete exploitation uses the vertex insertion probabilities in a deterministic manner and at each vertex-colour decision stage, allocation corresponds to the largest combined trail and desirability factor value. Meanwhile, exploration is achieved through stochastic implementation and vertex-colour decisions are obtained with reference to the roulette wheel. It was demonstrated that exploration enhances search capabilities.

To make ANTCOL a more attractive solution method, runtime conservation was considered. Some authors, Maniezzo (1999) for example, suggested that the RPR could be transformed into an additive linear function and thus, remove the computational expense of

power exponents. A similar philosophy was attempted here but the search lost its potency and poor solution quality resulted, which can be attributed to the difficulty of the examination timetabling problem. Therefore, alternative time saving implementations were tackled. Firstly, it has been noted that simply writing commands in different forms lead to quicker runtimes. Secondly, a series of values, a fixed length apart, were raised to a power, stored and read-in as a data file. This file acted as a look-up table. At each vertex-colour decision stage, each respective trail/desirability factor value was noted and the relevant interval in the look-up was pinpointed. Linear interpolation was then used to estimate that factor value when raised to some power. It was shown that the trade-off between time and solution quality was favourable. Thirdly, the choice of the next vertex was restricted to a subset of vertices to reduce the number of computations required of the RPR. This subset was defined as a candidate list and vertices were deemed eligible via some pre-defined criteria. It was observed that small candidate lists of vertices with better desirability scores not only reduced runtime but also improved solution quality. Fourthly, the storage of large trails slows the construction process and this is particularly noticeable as the search matures (handling larger trail scores). Imposing a suitable trail level maximum τ_{max} not only maintains solution quality but also reduces computational effort.

It has been demonstrated that the ant system is proving successful at finding feasible solutions and are comparable to other benchmark results. For example, our Best results are better than Merlot et al. (2002) – 18 for HEC, 24 for EAR and 21 for TRENT, and comparable with Carter et al. (1996) and Caramia et al. (2000) – 17 for HEC, 22 for EAR and 20 for TRENT.

Chapter 4

First-Order Conflict and Enhancement Techniques

4.1 Introduction

Chapter 3 provided insight into the capabilities of the ANTCOL algorithm and showed that the algorithm has the potential to be the basis of a good scheduling method. The biggest drawback lies with the computational effort that is needed and it has been demonstrated that savings can be made through amendments to the random proportional rule (RPR). In this chapter, alterations are made to the ANTCOL algorithm to enable application to the examination-scheduling problem and consequently, we redefine the name of the algorithm to AS-EXAM. However, firstly, ANTCOL is considered as an examination solution method. The objective of this study, referred to as the *Non-Determined Timeslot Case*, is to minimise the number of timeslots that are used. Secondly, extensions are made to the graph to allow the accommodation of additional constraints such as time-windowed and pre-assigned exams. In this study, the *Determined Timeslot Case*, timeslots are fixed according to the number pre-specified by the relevant institutions (refer to Table 3.1). In both cases, methods of solution enhancement are incorporated. Additionally, computational effort is discussed to determine the efficiency of an approach. Further data sets, in addition to HEC, EAR and TRENT, are used to illustrate the suitability of methods.

4.2 First-Order conflict

The main concern of any examination scheduler is to avoid the possibility of any student being required to sit two exams during the same sitting. This is known as the first-order conflict problem. To use a constructed timetable practically, there must be no such occurrence. Otherwise, the timetable is deemed as infeasible. Chapter 2 gave examples of literature, Broder (1964) and Mehta (1981) typically, that discussed techniques to tackle the first-order problem.

The first-order problem is an example of a hard constraint (those that cannot be violated). Another form of such a constraint is seating (room) capacity. A timetable is also classified infeasible if, for at least one timeslot, the number of students scheduled to sit exams exceeds the maximum seating capacity per timeslot for that institution. This binding

constraint is also detailed in Chapter 2. Examples of literature that discuss this topic are Lofti and Cerveny (1991) and Carter et al. (1994).

4.3 Non-Determined Timeslot Case

In some instances, although rare, institutions do not specify the number of timeslots that the exams need to be allocated to. Instead, minimising the number of timeslots may be the desired objective and is considered here. The aim of this section is to generate clash free timetables while minimising the number of timeslots. The basic algorithm (as described in Chapter 3) is used in addition to some modified versions of the basic algorithm that incorporate enhancement techniques – these are the standardisation of the reward function, elitism and hill climbing. These enhancement strategies are discussed below and related numerical results are presented in section 4.3.4.

4.3.1 Reward function

The role of the reward function is to quantify the perceived fitness of a constructed solution. This function passes quality information back to the trail and given that levels are instrumental in representing solution quality, the reward function itself is critical to the success of the method. Costa and Hertz (1997) suggested a reward function that is the inverse of the number of colours (nq) required to colour the graph $\frac{1}{nq}$. Chapter 3 showed that the ANTCOL algorithm performed well as a solution method. However, a drawback does exist. The range of nq scores varies between data sets. Therefore, it is harder to differentiate between an optimal and a near-optimal solution for some data sets. For example, let us consider two data sets, HEC and EAR. With HEC, an optimal solution is 17 colours, leading to a reward of 1/17, while a near-optimal solution could have 18 colours and a reward of 1/18. Here, the reward disparity is 0.00328. However, with EAR, an optimal solution is 22 colours, leading to a reward of 1/22, while a near-optimal solution could have 23 colours and a reward of 1/23. Here, the reward disparity is 0.00198. Taking this discussion one stage further, let us consider a problem with an optimal of 40 colours and a near optimal of 41 colours, the reward disparity is 0.00061. Therefore, the higher the number of colours required, the smaller the disparity between solutions, which may lead towards poorer solution quality due to the reduced ability to differentiate between solutions. This evidence suggests the need for reward standardisation between data sets.

Additionally, work presented later in this chapter will discuss the *Determined Timeslot Case* where examination timetabling problems have a prespecified number of timeslots. Therefore, alternative reward functions are needed to overcome the disadvantage of only colouring to the first *timeslot* colours. This will be discussed in further detail in Section 4.4.

Two proposed alternative reward functions are:

- $\frac{1}{(remain+1)}$ where *remain* refers to the number of vertices that cannot be allocated to the first *timeslot* colours. In the *Non-Determined Timeslot Case*, extra timeslots are created if necessary. However, since the number of timeslots is fixed in the *Determined Timeslot Case* unallocated exams are scheduled in the period which leads to the smallest number of direct clashes. Otherwise, such exams would not be subject to any trail increase, which is counter-intuitive, as unallocated exams should have a higher trail increase, therefore, making them more likely to be scheduled early. This is addressed later in the chapter.
- $\frac{1}{(nq-best+1)}$. This reward function interprets the difference between the number of colours required to colour the graph and the best-known solution, which for HEC, EAR and TRENT is equal to the size of the maximum clique.

Experiments have been performed that utilise the two proposed reward functions and will be made in direct comparison to the original reward function.

		HEC		EAR		TRENT	
Label	**Reward Function**	*Best*	*Av*	*Best*	*Av*	*Best*	*Av*
R1	$\frac{1}{nq}$	17	18.09	23	24.81	20	22.53
R2	$\frac{1}{(remain+1)}$	17	17.92	23	24.66	20	21.79
R3	$\frac{1}{(nq-best+1)}$	17	17.93	23	24.74	20	21.89

Table 4.1 – Average and best solution quality for proposed reward functions R1, R2 and R3.

The results confirm the intuition behind the investigation. Using a standardised reward function does generate better results than the raw counterpart, R1. Figures 4.1-4.3 illustrate the average solution quality versus time (cycle) relationship for the three proposed reward functions R1, R2 and R3 for HEC, EAR and TRENT respectively. These Figures indicate that while the finishing solutions of all reward functions for HEC and EAR are similar across all reward functions, the performances of R2 and R3 during the infancy of the searches are superior, especially for HEC. Meanwhile, the overall solution capabilities of R2 and R3 are considerably superior for the TRENT data set. It is observed that the performance of R2 and R3 are similar across all data sets with R2 generating marginally better results.

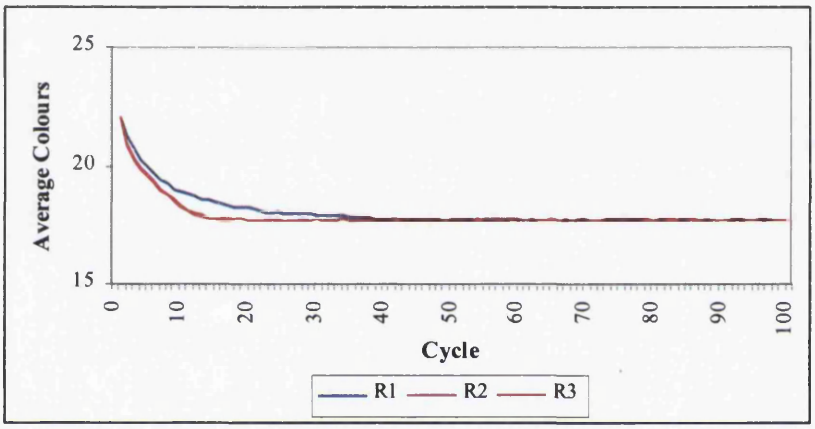


Figure 4.1 – Average Colours versus Cycle Plots for Proposed Reward Functions R1, R2 and R3 for HEC.

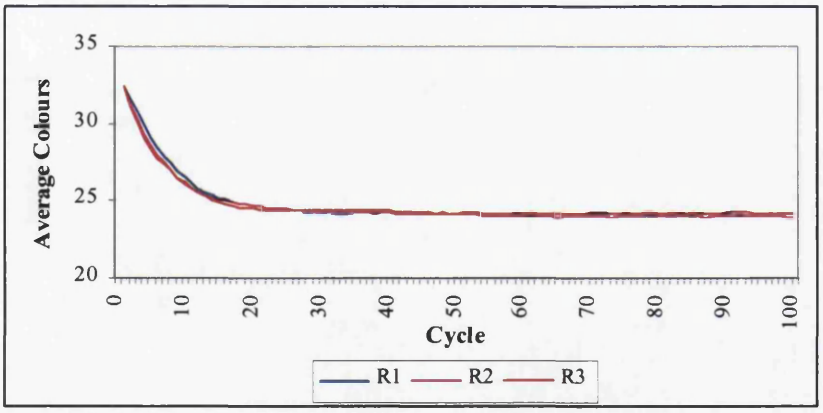


Figure 4.2 – Average Colours versus Cycle Plots for Proposed Reward Functions R1, R2 and R3 for EAR.

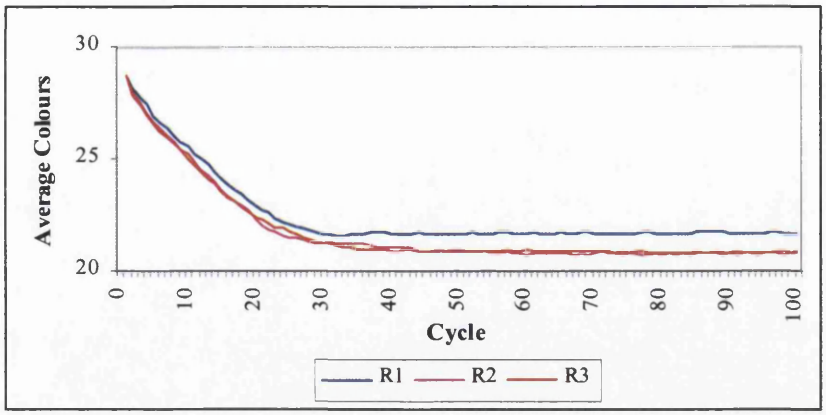


Figure 4.3 – Average Colours versus Cycle Plots for Proposed Reward Functions R1, R2 and R3 for TRENT.

This section has shown that some additional benefit can be obtained through using a standardised reward function and the reward function R2 will be used in all future experiments.

4.3.2 Elitism

Dorigo et al. (1991) showed that strengthening trails belonging to good solutions encouraged better overall solution quality (see Chapter 2 for description). During a run, a permanent copy of the 'elite' solutions is stored and extra pheromone replenishment is allocated to the trails associated with these solutions. It is believed that components of the 'elite' solutions could feature in the optimal solution and consequently, it is desirable to increase the probability of such components being selected throughout the run. The update philosophy can be described as follows.

$$\tau_{ij}(t+1) = \rho.\tau_{ij}(t) + \Delta\tau_{ij} + \Delta\tau_{ij}^{*} \quad (4.1)$$

$$\text{where } \Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \text{ and } \Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{L^k} \text{ if ant } k \text{ travels on edge } (i,j) \\ 0 \text{ otherwise} \end{cases} \quad (4.2)$$

$$\text{and } \Delta\tau_{ij}^{*} = \begin{cases} \sigma.\frac{Q}{L^*} \text{ if edge}(i,j) \text{is part of the best solution found} \\ 0 \text{ otherwise} \end{cases} \quad (4.3)$$

where $\Delta\tau_{ij}^{*}$ is the increase of trail level on edge (i,j) caused by the elitist ants
σ is the number of elitist ants

L^* is the tour length of best solution found
Q is a fixed constant

No precursory investigation is performed to obtain the appropriate parameter settings (σ and Q) since elitism will feature heavily in later parts of the thesis. One copy of the best global solution will be kept related experiments for the *Non-Determined Timeslot Case* (results displayed in Section 4.3.4).

4.3.3 Hill Climbing

Hill Climbing (*Hill*) is used to improve solution quality after solution construction has been completed. It accepts only improving moves, hence the name, and terminates when no further improvements are possible. *Hill* is performed as follows. Let us consider a timetable with nq colours (timeslots). The exams, e_r, allocated to timeslots numbered $best+1$ to nq, where *best* is the best-known solution, are stored in L_e. Each $e_r \in L_e$ is originally allocated to a timeslot m. The number of neighbours in each timeslot t for each exam e_r is evaluated, subject to $t<m$. The e_r/t combination with the lowest direct clashes that allows the reallocation of each neighbour in timeslots numbered less than $m-1$ is selected. Thus, e_r is inserted into t and the neighbours of e_r are inserted into other identified timeslots. In the event of a tie, a random allocation is made. After the relevant insertions, L_e is updated and the next exam is selected. If at any stage timeslot nq becomes empty then $nq=nq-1$. *Hill* terminates when no improving is found. Hill Climbing is applied to each *feasible* timetable.

4.3.4 Summary of results

All the above techniques along with *Amalgam CD*, which mixes heuristics *C* (*90%* of ants) and *D* (*10%*) in each cycle as proposed in Section 3.5, have been considered. These methods will be compared against single pass heuristic *C*, the basic ANTCOL algorithm and benchmark examination timetabling heuristics – Carter et al. (1996), Caramia et al. (2000) and Merlot et al. (2002). Details regarding the more specific nature of these algorithms have been given in Chapter 2. Elitism and Hill Climbing are applied as discussed in Sections 4.3.2 and 4.3.3. To fully assess the success of the proposed methods, a wider range of data sets are considered.

Since the stated authors have not used the Swansea data sets, it was felt reasonable to exclude them from consideration here (but will feature later). Table 4.2 presents the best solutions found.

Data Set	*Construction Heuristic C*	*Basic*	*Amalgam CD*	*Elitism*	*Hill*	*Carter et al.*	*Caramia et al.*	*Merlot et al.*
C91	42	29	**28**	29	**28**	28	28	30
C92	39	**28**	**28**	**28**	**28**	28	28	31
EAR	28	23	**22**	**22**	**22**	22	22	24
HEC	19	**17**	**17**	**17**	**17**	17	17	18
KFU	21	**19**	**19**	**19**	**19**	19	19	21
LSE	19	**17**	**17**	**17**	**17**	17	17	18
RYE	24	22	**21**	22	**21**	21	21	22
STA	**13**	**13**	**13**	**13**	**13**	13	13	13
TRENT	24	**20**	**20**	**20**	**20**	20	20	21
UTA	42	31	31	31	31	32	30	32
UTE	**10**	**10**	**10**	**10**	**10**	10	10	11
YOR	23	21	20	20	**19**	19	19	23

Table 4.2 – Best statistics for Graph Colouring, Basic Ant System, Basic Ant System with Enhancement Techniques and Benchmark Results.

These statistics prove that the basic ant algorithm (*Basic*) outstrips the single pass heuristic and again, emphasises that the ant algorithm does improve solution quality. The bolded figures note when the best-known solution has been reached. *Basic* performs well in direct comparison to the benchmark heuristic methods. Results are very competitive for 7 of the data sets [C92, HEC, KFU, LSE, STA, TRENT and UTE], but the *Basic* does not find the best-known result with C91, EAR, RYE, UTA and YOR. Mixing construction heuristics *C* and *D* in each cycle does improve solution quality above *Basic*. Out of the 5 data sets where the optimum was not found, 3 of them are discovered through *Amalgam CD* [C91, EAR and RYE]. Meanwhile, both *Amalgam CD* and *Elitism* reduce the best result for YOR by the same measure, drops from 21 to 20 colours. The most successful enhancement is *Hill*, which enables the best-known result to be found with all data sets except UTA.

Overall, the results are pleasing and indicate that ANTCOL, with the aid of some enhancements, can be taken seriously as a basic examination-scheduling tool on an uncapacitated first-order level when the aim is to minimise the number of timeslots.

4.4 Determined Timeslot Case

In this section, we discuss the *Determined Timeslot Case*. Each exam data set comes equipped with a predetermined number of timeslots that must feasibly accommodate all the exams. In this section, we extend the graph to accommodate timeslot vertices. This

provides the scheduler with greater flexibility and can accommodate side constraints such as pre-assigned and time-windowed exams. Edges inserted between an exam vertex i and a timeslot vertex j indicates conflict, therefore i cannot be allocated to j. For example, if an exam i is pre-assigned to timeslot j, the graph can accommodate this by forming conflict between the exam vertex i and all timeslot vertices other than j. Additionally, the timeslot vertices form a clique, which allows only one timeslot vertex to be accommodated within each colour group. Balakrishnan (1991) detailed the extension of the graph while using the Recursive Largest Degree First, see Leighton (1979) and Appendix 2.1, graph colouring based heuristic to schedule exams at the Freeman School of Business at Tulane.

The idea of including timeslot vertices can be problematic if distinction is not made between exam and timeslot vertices. Dowsland et al. (2002), detailed in Appendix 4.1, discussed that, in some cases, the degree of some exam vertices may be larger than timeslot vertices if there are a small number of pre-assignments and time-windows. In theory, the allocation of higher degree vertices earlier in the construction process increases solution quality. However, a comparative analysis was performed that proved that selecting timeslot vertices first per colour group, irrespective of degree, produces more feasible results than not. Experiments were performed on two real-life data sets, Swan2000 and Swan2002 (description of data sets can be found later in this chapter) and it was found that Swan2000 had a feasibility rate of 23.20% when the first vertex to be coloured was not certain of being the timeslot vertex and when the first vertex in each colour group had to be a timeslot vertex feasibility improved to 95.20%. For Swan2002, the feasibility rate jumped from 47.20% to 93.60% for the same comparison.

Timeslot vertices can be selected randomly or according to degree. Since it has been shown that randomness plays an important part in ant systems, it is appropriate to stipulate that each timeslot vertex is chosen on a random basis.

Due to the structural change of the graph, it seems advisable to observe the influence of the two major determinants on solution quality (as discussed in Chapter 3) - bias parameters and construction heuristic. For each construction heuristic, the timeslot vertex is chosen randomly. For heuristics A, B and C the first exam vertex is selected random-proportionately (biased according to trail), while it is chosen deterministically (according to maximum degree) for D, E and F. Appendix 4.2 presents the results for $\alpha, \beta \in [1, 2, 3,$

4] and the eight construction heuristics. These results still conform to the similar study that was performed in Chapter 3. The strongest construction heuristic is *C*, while similar findings to Chapter 3 regarding the bias parameters are observed. Increasing α encourages the production of more feasible timetables, while β has a similar influence. As before, the bias parameter α is a critical factor with respect to solution quality and stipulating $\alpha \geq 2$ is necessary to produce the desired volume of feasible timetables and the (general) improvement in solution quality slows down across β as α gets bigger.

Each ant will construct timetables up to the number of timeslots that are specified. Any unallocated exam will be placed into the timeslot that creates the minimum number of direct conflicts. Such practice is necessary for the work detailed in the next chapter to estimate the second-order score of an infeasible timetable. However, as mentioned earlier, it does create a problem regarding trail updates. Each pairwise exam set, and those involving direct clashes, is updated according to reward function R2. Solution characteristics belonging to infeasible timetables are penalised collectively, however no individual penalties are placed on the unallocated exams. This is counter-intuitive, as we intend to make these 'difficult to colour' vertices more attractive so that they are coloured earlier in the process, before being blocked by their more appealing neighbours. The results presented in Appendix 4.2 indicate that feasibility is maintained despite this obvious drawback. However, work presented in Section 4.4.1.3 demonstrates that allocating additional trail to the unallocated exams increases solution quality.

4.4.1 Solution enhancement

When using the algorithm in its basic form, the standard of solution quality is good but it is possible to improve solution quality further through enhancement methods. The simplest approach to enhance solution quality is to increase the bias exponents on the trail and visibility factors in the RPR function. By imposing $\alpha=4$ and $\beta=1$ typically, the rate of feasible timetables jumps from 86.58 (rate per 100) to 94.62 timetables, 81.32 to 94.01 timetables and 86.59 to 95.81 timetables for HEC, EAR and TRENT respectively. These statistics are lifted from Appendix 4.2. Many of these extra feasible timetables contain near-identical characteristics but the production of reoccurring optimal solutions is still a reflection of the algorithms ability to generate the desired solutions (a study on the diversity of solution characteristics can be found in Section 4.4.1.3). Another proven

method of enhancing first-order solution quality was *Candidate Lists* as discussed in Chapter 3. Lists with constitutions biased to desirability scores increase feasibility rates. Typically, lists with ten members, ordered according to desirability generate average *Feasible* of 88.98, 88.26 and 96.02 for HEC, EAR and TRENT respectively, which in all cases are superior to the *Basic* system. It is demonstrated in Chapter 3 that the incorporation of particular lists reduces runtimes due to the reduced number of probability calculations performed by the RPR.

The work performed in conjunction with time saving procedures in Chapter 3 highlighted the computational expense of increasing bias. In this section, we discuss alternative approaches to solution quality enhancement while attempting to make it attractive with regard to computational effort.

4.4.1.1 Basic Trail Reinforcement (*BTR*)

In this section, a relatively basic form of elitism is introduced. After the production of each feasible solution, the trails associated with the solution are reinforced by some constant, *rc*. The aim is to bias future ants to select pairwise exam sets that feature within a feasible solution, thus, (potentially) leading to an increased proportion of feasible solutions being generated. Negatively, such a method depends on the generation of feasible timetables before trail reinforcement can be performed, but positively, the ant algorithm generally constructs feasible timetables with relative ease and the runtimes associated with such experiments differ minimally from the basic algorithm.

	HEC		EAR		TRENT	
rc	*Unallocated*	*Feasible*	*Unallocated*	*Feasible*	*Unallocated*	*Feasible*
1	0.29	86.58	0.83	81.32	0.68	86.59
2	0.26	88.52	0.81	81.89	0.66	87.30
3	0.26	88.22	0.79	83.82	0.63	87.85
5	0.23	89.79	**0.76**	**86.21**	0.62	87.48
10	0.24	88.45	0.75	85.72	**0.58**	**87.96**
20	0.23	88.61	0.82	80.34	0.60	85.69
30	**0.20**	**90.92**	0.79	82.17	0.62	84.00
50	0.22	88.90	0.80	81.20	0.63	82.88

Table 4.3 –Unallocated and Feasible for Range of rc.

Table 4.3 presents the average number of unallocated exams per timetable and the average number of feasible timetables per 100 (*Feasible*) for an experimental range of *rc*. When *rc=1*, no additional reward is given and will act as a comparison against alternative *rc*. The

values bolded refer to best *Feasible* for each data set. It appears that incorporating a basic form of trail replenishment enhances the number of feasible timetables that are generated, while, in some cases, allowing large trail replenishments actually lowers solution quality. Imposing strong trail replenishment reduces the exploratory power of the ants as they become biased towards certain constructions and consequently, the ants have an inability to explore sufficiently to find alternative feasible constructions, thus lowering the overall number of feasible solutions obtained. For HEC, the number of feasible solutions increases from 86.58 to 90.92 when rc=30. For EAR, feasible solutions raises from 81.32 to 86.21 when rc=5 and for TRENT, 86.59 to 87.96 when rc=10.

There is no discernable pattern for rc across the data sets. However, it can be stated that solution quality improves as rc increases for relatively small rc. Solution quality then maximises for some rc and solution quality eventually deteriorates for relatively large rc. The preferred values of rc here varies between data sets but a good compromise setting for all data sets is rc=5.

4.4.1.2 Elitism

A more sophisticated form of rewarding superior solutions is *Elitism*, Dorigo et al (1991). The method has been touched upon earlier in this chapter and a slight variant of elitism will be used here (see Section 5.5.3 for further discussion) to improve the volume of feasible solutions that are generated. In this section, two factors will be examined – the number of elitist ants e and a weight component w, such that each elite solution is magnified by w. A copy of the e best solutions are stored and used to reinforce the trails at the end of each cycle during a run of the algorithm. During a search, the e best solutions s are ordered $s_1, s_2, \ldots, s_e$ according to fitness criterion. Each newly observed solution s_n is compared against the fitness of the lowest ranked elite solution s_e. If s_n is deemed superior to s_e then s_n replaces s_e in the elite list and the process continues. Firstly, experiments are performed to deduce sensible settings of e when w=1 (one extra copy per solution), and secondly, the influence of w is examined and e is chosen according to the findings previously presented.

Results for various e are presented in Table 4.4.

	HEC		EAR		TRENT	
e	*Unallocated*	*Feasible*	*Unallocated*	*Feasible*	*Unallocated*	*Feasible*
0	0.29	86.58	0.83	81.32	0.68	86.59
1	0.29	87.00	**0.76**	**85.28**	0.66	86.52
2	**0.28**	**87.97**	0.75	84.74	**0.66**	**86.93**
3	0.30	85.36	0.75	84.91	0.68	86.42
5	0.30	85.73	0.85	84.79	0.69	86.20
10	0.31	84.76	0.76	85.17	0.69	85.40
15	0.29	86.53	0.77	84.10	0.73	84.78
20	0.33	84.32	0.79	84.05	0.76	83.00
30	0.36	82.30	0.86	81.90	0.76	82.92

Table 4.4 - Unallocated and Feasible for range of e.

The benefit of replenishing trails with a set of elite solutions seems limited, however it is noted that $e>0$ is superior to $e=0$. The best results are discovered for low e. Generally, allowing large e does not encourage enough differentiation between solutions and consequently, the exploitation capabilities of the ants are reduced. However, the disparity in solution quality across e is insignificant. Now we test the influence of w. For each experimental w, e is set at 2, as this setting produced the general best solution across the data sets in the previous study.

	HEC		EAR		TRENT	
w	*Unallocated*	*Feasible*	*Unallocated*	*Feasible*	*Unallocated*	*Feasible*
1	0.28	87.87	**0.75**	**84.74**	0.66	86.93
2	**0.27**	**88.30**	0.77	84.60	**0.65**	**87.32**
3	0.27	87.99	0.77	84.04	0.69	86.09
5	0.31	84.29	0.83	78.99	0.75	82.11
10	0.33	82.70	1.01	67.49	0.89	80.52
15	0.44	75.10	1.17	61.10	0.93	78.24
20	0.64	61.29	2.13	24.32	1.02	68.34
30	1.10	44.85	1.57	46.76	1.63	52.98

Table 4.5 - Unallocated and Feasible for Range of w.

Minimal benefit can be achieved by setting $w>1$ i.e. incorporating more than one additional copy of an elitist solution. Results indicate that larger w leads to poor solution quality. During the infancy of the search, higher w encourages the ants to learn quicker and consequently, solution quality improves at a faster rate for large w. However, large w will have a saturation effect and high levels of trail are laid covering approximately e solutions. After some stage, ants become trapped into producing the same subset of solutions and this is represented by the inability to produce a greater level of feasible solutions after a certain stage of the run. The *Average Unallocated* rate across five independent runs for HEC were averaged and plotted (in Figure 4.4) against *Cycle* for two experimental values of w. This is used to illustrate the problems experienced for larger w.

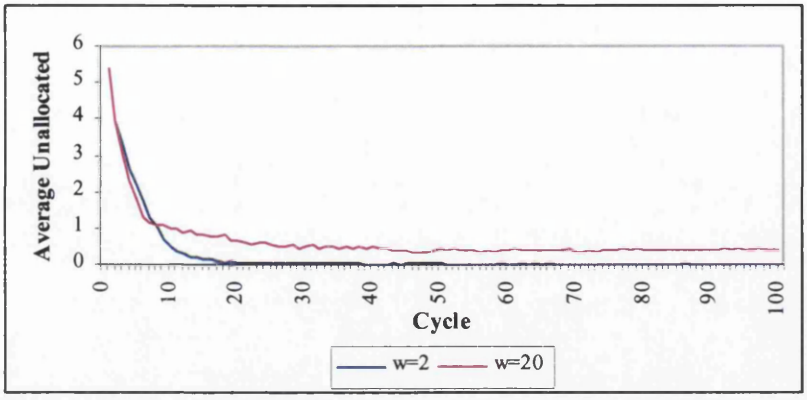


Figure 4.4 – Average Unallocated Exams per Timetable versus Cycle Plots for w=2 and w=20 for HEC.

This section has been informative since it indicates the detrimental influence of placing too much emphasis on better trails. However, the results achieved here have been disappointing and do not compete with the less sophisticated *BTR* technique as detailed in section 4.4.1.1.

4.4.1.3 Problematic Exam Trail

Construction heuristics attempt to insert potentially more difficult vertices/exams into solutions earlier in the construction process e.g. it is deemed logical to insert vertices of higher degrees as early as possible. However, heuristics do not have the ability to learn and the use of the trail aids the construction process by biasing the ants to insert exams at particular exam-timeslot decision points. Here, we combine the characteristics of trail and construction heuristics in order to improve the capabilities of the solution construction process further.

Until this juncture, after an ant has constructed a timetable, exams not allocated in the prespecified number of timeslots are inserted into the timeslots that create the minimum direct clashes. All pairwise exams are then subjected to trail updates according to the reward function R2. Therefore, timetable characteristics are penalised collectively. Here, in addition to normal trail updates, exams are penalised individually in order to quantify difficulty of allocation. If the same exams are left unallocated over a series of timetables, concerted attempts need to be made to build constructions around these exams to ensure feasibility. We introduce *Problematic Exam Trail (PET)*, which can be described as an intelligent diversification function that guides the search away from solutions in which the

previously uncoloured vertices fail to be coloured by increasing their probability of being chosen.

To avoid confusion within this section, the exams that could not be accommodated within t timeslots due to hard constraint violations will be defined as *problem* exams. Those exams that are not (yet) accommodated within the current partial solution (i.e. not all timeslots have been constructed) will be classed as *unallocated.*

PET introduces a trail matrix which stores the 'problematic' trail between a *problem* exam u and all other exams v. For an exam u, high trail levels indicate the need for early insertion during the construction process.

Let $s_1, s_2,\ldots,s_{Na}$ be the timetables obtained during a cycle and let $R2_a$ (see Section 4.3) represent the fitness of timetable s_a. After the completion of each infeasible timetable, let the set U store the *problem* (those not allocated) exams and the set V store the *allocated* exams. The values PET_{uv} in *PET* are updated as follows.

$$\Delta PET_{uv} = \Delta PET_{uv} + R2_a \qquad \forall u \in U,\ \forall v \in V \quad (4.4)$$

After the completion of each cycle, the *PET* matrix is subjected to an evaporation parameter, $(1-\rho_2)$, in order to accommodate potential construction changes during the search.

$$PET_{uv} = \rho_2 . PET_{uv} + \Delta PET_{uv} \quad (4.5)$$

Meanwhile, the RPR is altered to accommodate $PET(s[k-1],i)$ which quantifies the level of 'problematic' trail. Let the set W contain the *unallocated* but feasible exams.

$$p(k,i,j) = \frac{\tau(s[k-1],i,j)^{\alpha} . \eta(s[k-1,i])^{\beta} . PET(s[k-1],i)}{\sum_{r \in W} \tau(s[k-1],i,r)^{\alpha} . \eta(s[k-1],i,r)^{\beta} . PET(s[k-1],i,r)} \quad (4.6)$$

The *PET* term is used as a coefficient and is not raised to a bias parameter and can be calculated as follows.

$$PET(s[k-1],i) = \sum_{v \in \Omega} PET_{iv} \quad (4.7)$$

where Ω is the set of all other exams.

Experiments have been performed for the main three data sets across a range of ρ_2.

	HEC		EAR		TRENT	
ρ_2	*Unallocated*	*Feasible*	*Unallocated*	*Feasible*	*Unallocated*	*Feasible*
None	0.29	86.58	0.83	81.32	0.68	86.59
0.01	0.21	89.38	0.49	87.09	0.50	88.44
0.10	0.19	90.90	0.42	89.82	0.29	91.07
0.30	0.17	91.34	0.38	91.61	0.25	93.10
0.50	0.16	92.81	0.36	92.10	0.23	93.87
0.70	0.16	92.45	0.35	92.81	0.22	94.11
0.90	0.16	93.08	0.35	92.49	**0.18**	**94.90**
0.99	0.15	93.26	0.33	92.95	0.18	94.50
1.00	**0.14**	**93.38**	**0.32**	**93.19**	0.19	94.48

Table 4.6 - Unallocated and Feasible for Range of ρ_2.

The general relationship between ρ_2 and solution quality is clear. Solution quality improves as ρ_2 increases. Forcing the early introduction of *problem* exams has a positive influence. Even the allowance of minimal *PET* information (i.e. when $\rho_2=0.01$) enhances results. However, stipulating a generic setting of $\rho_2=1$ appears beneficial. The extra effort required above *Basic* is minimal and this is demonstrated by the runtimes presented in Section 4.4.1.5.

To evaluate the diversity of the distribution of vertex-colour/exam-timeslot assignments over a population of solutions entropy values, Fleurent and Ferland (1996), values (as described in Equations 4.8 and 4.9) could be used.

$$D_i = \frac{-\sum_{j=1, n_{ij} \neq 0}^{t} \left(\frac{n_{ij}}{P}\right) . \log\left(\frac{n_{ij}}{P}\right)}{\log t} \quad (4.8)$$

$$D = \frac{\sum_{i=1}^{nv} D_i}{nv} \quad (4.9)$$

where t is the number of timeslots, nv is the number of exams, P is the number of feasible solutions and n_{ij} is the number of solutions in P that exam i is allocated to timeslot j. Entropy values are normalised in the interval [0,1]. A population of ants returning identical solutions will generate an entropy value of 0, while a uniform distribution will have a value of 1. The entropy values and *Feasible* rates (both averaged per run) for each of the experimental runs is presented in Table 4.7.

	Entropy		**Feasible**	
	Basic	*PET*	*Basic*	*PET*
HEC	0.051	0.051	85.97	94.76
EAR	0.042	0.044	76.64	92.50
TRENT	0.095	0.096	87.13	94.42

Table 4.7 –Entropy and Feasible

Table 4.7 demonstrates that the entropy values for both systems are close to 0 for the data sets. This suggests that many solutions have similar characteristics. Experiments when using AS-EXAM often converge (exploitation) to a set of solution characteristics and convergence is detected relatively early (as discussed in Chapter 3) in the allocation of 100 cycles. Therefore, the entropy values will be lower as a result. Table 4.7 shows that the disparity in entropy values between *Basic* and *PET* is minimal and this indicates that the introduction of *PET* does not lower solution diversity, but rather guides the search away from solutions that fail to colour certain exams.

A benefit of *PET* is the construction of feasible timetables earlier in the search. Table 4.8 presents the absolute difference of the cycle numbers when the first feasible timetable was observed between the *Basic* and *PET* systems (the results in parentheses are the actual cycle numbers, with the first value relating to the *Basic* system). Each independent run is considered.

	Experimental Run				
	1	*2*	*3*	*4*	*5*
HEC	1 (3-2)	1 (4-3)	1 (3-2)	0 (3-3)	0 (2-2)
EAR	6 (10-4)	7 (11-4)	6 (10-4)	4 (9-5)	7 (10-3)
TRENT	7 (9-2)	6 (8-2)	8 (10-2)	4 (7-3)	6 (9-3)

Table 4.8 – Absolute Difference of Cycle Numbers when First Feasible Solution was observed between Basic and PET Systems.

Table 4.8 shows that feasible timetables are achieved earlier in the search through the *PET* system. This demonstrates that the benefit of *PET* on the search is almost immediate.

Gambardella et al. (1999a) proposed a similar concept to *PET* with respect to the Vehicle Routing Problem with Time Windows (VRPTW). A detailed account can be found in Chapter 2. In VRTPW, the desirability η_{ij} is calculated by taking into account the travelling time t_{ij} between locations i and j, the time window associated with location j and a vector that stores the number of occasions that location j has not been inserted into the solution. This latter component is the one of interest.

4.4.1.4 Hill climbing *(Hill)*

Methods that influence trail levels have been introduced in order to enhance solution quality. In this section, *Basic* is used with no trail enhancement strategies and improvements to timetables are performed through a simple hill climbing strategy, which differs slightly from the approach taken in Section 4.3.3. After the completion of each non-feasible timetable, attempts are made to schedule the unallocated exams within the first *timeslot* colours with the reallocation of neighbours being subject to the same restriction. *Hill* continues until no feasible insertions are viable.

	HEC		EAR		TRENT	
Strategy	*Unallocated*	*Feasible*	*Unallocated*	*Feasible*	*Unallocated*	*Feasible*
No Hill	0.29	86.58	0.83	81.32	0.68	86.59
With Hill	0.11	94.88	0.39	89.55	0.30	93.96

Table 4.9- Unallocated Exams and Average Feasible for Basic and Hill.

It is clear that the use of *Hill* does generate better results than *No Hill (Basic)*, but the trade-off is the additional runtime that is required (See Section 4.4.1.5). Reducing the number of timetables that are improved would naturally lower runtimes since less use of *Hill* is required. However, this section intends to illustrate the solution quality benefit of certain implementations when no usage restrictions are enforced.

Based on the results presented in Sections 4.4.1.3 and 4.4.1.4, it is shown that the *PET* philosophy generates better solutions than *Hill* while requiring less runtime. However, the efficiency of *PET* depends on the quality of the solutions generated by the basic algorithm. The presence of many unallocated exams will result in the update of numerous trails in the *PET* matrix. Since the aim of *PET* is to identify a subset of *problem* exams, limited differentiation between exams will occur and the benefit of *PET* will be reduced. Experiments were performed for a less efficient ant based system. When $\alpha=1$ and $\beta=1$ for

HEC, runs of *Basic* produced an average unallocation rate of 4.06 exams and an average production of 0.14 feasible timetables per cycle. Incorporating the *PET* technique, with no evaporation, did improve the standard of solution quality, but not significantly. The unallocation rate drops to 2.61 with a feasible score of 3.74 timetables per cycle. This evidence shows the importance of a high quality basic construction process so that the *PET* concept can be beneficial. Meanwhile, the *Hill* procedure fares better (which is expected) under the same bias parameter conditions. Due to the allocation of fewer exams, the direct conflict scores per timeslot will be lower. Consequently, the placement of unallocated exams and reallocation of neighbours is easier during the early stages of *Hill*, although, insertions are more difficult as time progresses. However, the greater flexibility of this method will lead to the production of more feasible timetables and a lower unallocation rate. For comparison, the *Hill* strategy improved the basic system from an unallocation rate of 4.06 to 1.82 exams and the volume of feasible timetables per cycle increases from 0.14 to 43.58.

4.4.1.5 Summary of Results

With respect to solution quality, the superior improvement techniques are *PET* and *Hill*. These approaches will be used when other members of the data pool are considered here (see Chapter 3 for data pool information). Additionally, a combined approach of *PET* and *Hill* is implemented. The reasoning for this amalgam is as follows. *PET* has been shown to enhance solution quality considerably when good parameters are chosen for the algorithm itself while *Hill* also performs well but requires additional runtime. After the construction of a timetable, if it is classified as infeasible, it will be subject to the improvement strategy *Hill*. Since *Hill* will attempt to insert fewer exams into the constructions due to the goodness of *PET*, the runtimes will become smaller and the volume of feasible timetables will increase. The average *Feasible* timetable rate per 100 timetables and average *Time* (in seconds per experimental run) for the four methods are presented below.

	BASIC		PET		HILL		PET/HILL	
Data Set	*Feasible*	*Time*	*Feasible*	*Time*	*Feasible*	*Time*	*Feasible*	*Time*
C91	84.06	6851.30	94.13	6690.01	91.39	40488.50	96.19	14121.30
C92	83.97	3891.51	93.70	3523.25	93.91	17524.80	95.87	6851.86
EAR	76.64	188.23	92.50	184.89	89.54	410.25	95.33	236.91
HEC	87.13	35.69	92.42	35.46	94.88	45.98	96.75	38.96
KFU	83.70	1622.72	96.94	1595.88	94.58	3865.17	98.49	1798.58
LSE	78.90	839.42	96.07	823.73	91.71	2359.18	97.55	979.55
STA	95.63	98.71	98.92	95.56	98.91	102.17	99.43	98.51
TRENT	85.97	296.59	94.76	290.84	93.99	671.13	97.61	374.06
UTA	83.58	5721.80	94.65	5574.11	96.17	34908.10	96.56	10576.90
UTE	60.77	149.90	95.21	146.05	88.88	224.92	98.24	157.66
YOR	52.72	169.96	89.10	169.65	81.54	473.55	93.77	234.99

Table 4.10 –Feasible and Runtimes for Extended Number of Data Set comparing Enhancements.

Introducing some form of enhancement technique has a positive influence on solution quality. These strategies increase the number of feasible timetables for each of the data sets. Note the vast improvement experienced with YOR. This is a particularly difficult data set and the benefit of some form of enhancement technique is obvious. It is also shown that the *PET* technique is generally marginally superior to *Hill* but the runtimes are non-comparable. The computational times recorded for *PET* are insignificantly different from *Basic*. Since the runtime disparity between *Basic* and *Hill* is large, the selection of *PET* as a solution improvement technique is logical, particularly when used in conjunction with larger data sets. Additionally, the amalgam of *PET* and *Hill* improves solution quality further, while requiring notably shorter average runtimes than *Hill*. The reasoning for efficient runtimes is a consequence of *PET* reducing the number of problem exams and thus, less use of *Hill* is required.

4.5 Determined Timeslot Case - Capacitated

Uncapacitated examination timetabling problems are not realistic since institutions do not have an unlimited amount of seating capacity. Consequently, the amount of total room space must be a binding constraint since it cannot be violated. Burke et al. (1995a) stated in their survey paper that 94% of institutions regard seating constraints as their biggest concern (when direct clashes are not considered). In addition, institutions often impose other criteria such as non-split (students sitting the same exam must be allocated to the same room) exams and one exam per room, which all contributes to make scheduling an examination timetable even more difficult. Burke et al. (1995c) and Burke et al. (2000) presented memetic and multistage approaches respectively to the capacitated examination-

timetabling problem and generated timeslot capacities for some of the data sets used here. A capacity was created by dividing the total number of students sitting the exams by the number of sessions and adding 5% slack, Merlot et al. (2002). This rule is applied to all the data sets. The capacities are as in Table 4.11.

Data Set	*Capacity*
C91	1550
C92	2000
EAR	350
HEC	650
KFU	1995
LSE	635
STA	465
TRENT	655
UTA	2800
UTE	1240
YOR	300

Table 4.11 – Timeslot capacities for each data set

Within the AS-EXAM algorithm, no formal part of the algorithm encourages solutions that satisfy room constraints. Allocations are not allowed if they break the room constraints. The structure of Section 4.4.1.5 is repeated here, but for capacitated problems. *PET* maintains the same structure, while *Hill* is modified slightly to restrict exam moves that would violate the timeslot capacities. The *PET-Hill* amalgam is again tested. These approaches are compared against the *Basic*, non-enhancement, algorithm. The results are as follows.

	BASIC		**PET**		**HILL**		**PET-HILL**	
Data Set	*Feasible*	*Time*	*Feasible*	*Time*	*Feasible*	*Time*	*Feasible*	*Time*
C91	0.00	7361.50	0.00	7825.91	0.49	103356.00	0.03	82132.90
C92	59.98	4669.08	86.96	4679.72	77.34	29150.80	90.16	11738.70
EAR	0.00	325.24	53.43	352.48	2.92	1657.41	58.05	907.11
HEC	0.00	63.56	0.00	77.80	0.20	250.32	0.10	263.19
KFU	81.54	1430.64	96.75	1852.46	93.03	4502.63	98.30	2067.43
LSE	29.85	1539.29	71.44	1606.64	42.72	6219.69	84.51	3113.86
STA	15.56	135.52	26.38	141.62	16.98	366.12	38.45	304.51
TRENT	0.26	430.64	21.48	448.46	2.02	2148.74	29.89	2470.92
UTA	79.37	6281.35	93.44	6369.14	87.65	72134.70	95.98	15326.51
UTE	25.48	123.62	64.30	125.09	34.55	276.55	70.64	186.75
YOR	0.00	209.64	35.71	220.19	6.78	1046.33	45.63	721.40

Table 4.12 – Feasible and Runtimes for Extended Number of Data Sets for Capacitated Model comparing Enhancements

These experiments highlight the benefit of enhancement techniques and illustrated the potency of the trail based *PET* approach. This method performs consistently better than *Hill*. This stresses the benefit of an efficient construction process rather than simply

obtaining complete solutions and then attempting to deal with the problematic exams via exam moves. These capacitated problems are considerably more constrained than their uncapacitated counterparts. Solution qualities associated with some of the data sets are not affected by timeslot capacity e.g. KFU and UTA, however, the majority are and emphasise the role of the enhancement techniques. However, all associated results of HEC and C91 are of unacceptable standard. However, the improvement observed with EAR does provide some compensation.

It should be noted here that capacitated-related experiments regarding five of the above data sets (with respect to the second-order problem) have been performed by Merlot et al. (2002), Burke et al. (1995c), Caramia et al. (2000) and Di Gaspero and Schaerf (2000). In most instances, the number of timeslots correspond to the absolute upper limit specified by the institution and therefore, do not correspond to the related information presented in Table 3.1. For C91, 51 timeslots are used instead of 35, for C92, 40 slots are used in place of 32, for TRENT, 35 are used instead of 23 and for UTA, 38 timeslots are used in place of 35. The number of timeslots used for KFU remains unchanged. The extra timeslots used indicate the additional difficulty of observing capacities and therefore, go some way to explain the scheduling troubles with C91 and TRENT. Encouragingly, at least one feasible solution is returned for all data sets and relaxing the number of timeslots will enhance solution quality further.

Figures 4.5-4.6 illustrate the number of students per timeslot within a feasible timetable for two sample data sets, EAR and TRENT. It is observed that the number of students equals the timeslot capacity for many timeslots. For EAR, 13 timeslots (of 24) use the maximum seating capacity of 350 students. For TRENT, 20 timeslots (of 23) require the use of the upper limit of 655 seats. Additionally, high percentages of the seating capacities are used within all the other timeslots for both examination problems (EAR and TRENT). These diagrams emphasize the difficulties that exist when attempting to obtain clash-free timetables that observe tight (5% slack) seating capacities. Therefore, a method that is able to generate a stream of feasible timetables, such as capacitated AS-EXAM, should be highly regarded.

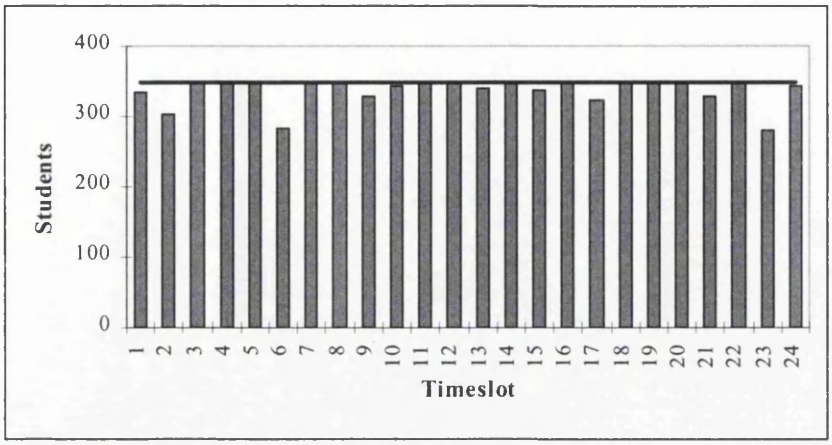


Figure 4.5 –An Example of Number of Students per Timeslot for a Feasible Timetable for EAR (Max 350 Students)

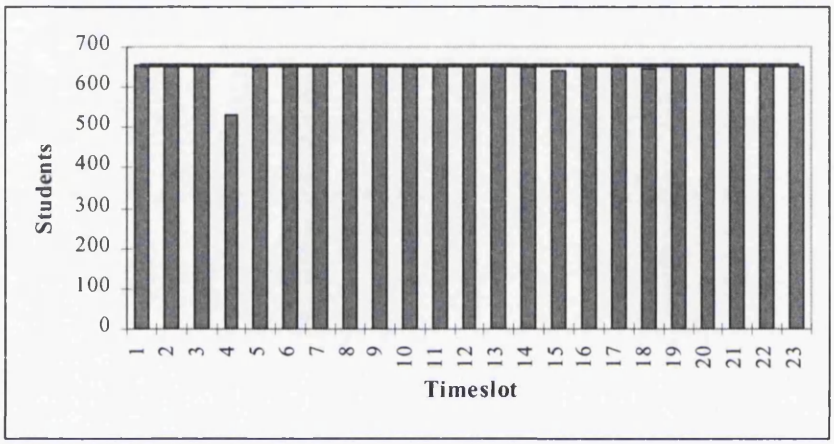


Figure 4.6 –An Example of Number of Students per Timeslot for a Feasible Timetable for TRENT (Max 655 Students)

4.6 Swansea data sets

Two University of Wales Swansea data sets are used within this thesis – January 2000 (Swan2000) and May 2002 (Swan2002). The descriptive statistics regarding these data sets are detailed in Chapter 3. Given that these data sets contain examples of the side constraints associated with real life examination timetabling problems, it was felt wise to apply some of the techniques introduced within this chapter to those Swansea data sets when regarding the *Determined Timeslot Case*.

Swan2000

This data set contains 313 exams that need to be scheduled within 20 timeslots. There are 14 preassigned and 20 time windowed exams. There are 22 simultaneous exam combinations.

Swan2002

This data set contains 722 exams that need to be scheduled within 34 timeslots. There are 27 preassigned and 47 time windowed exams. There are 148 simultaneous exam combinations.

The techniques applied to each instance are *Basic*, *PET*, *Hill* and *PET-Hill*. The studies relating to each Swansea data set are sub-divided according to whether seating capacities **(994 seats)** are observed (see Table 4.13). There is 20.24% and 13.11% seating capacity slack for Swan2000 and Swan2002 respectively.

		BASIC		PET		HILL		PET-HILL	
Data Set	**Capacity Observed?**	*Feasible*	*Time*	*Feasible*	*Time*	*Feasible*	*Time*	*Feasible*	*Time*
Swan2000	Y	70.74	405.68	98.83	406.82	73.09	873.81	99.15	424.88
	N	89.54	401.88	98.89	406.24	98.64	625.46	99.83	418.54
Swan2002	Y	45.42	4097.39	98.49	4114.10	46.69	6184.74	98.71	4524.15
	N	88.53	4090.70	98.93	4103.31	99.84	10235.60	99.99	4455.17

Table 4.13 -Feasible and Runtimes for Swansea Data Sets for Uncapacitated and Capacitated models comparing Enhancements.

Table 4.13 shows that *Basic* is able to produce good volumes of feasible timetables for data sets that come accompanied with time-windowed and pre-assigned exams. This also applies when seating capacities need to be observed. Table 4.13 again demonstrates the capabilities of the *PET* trail with respect to improved solution quality and with minimal extra computational effort. Improved feasibility rates are possible through *Hill*, but these are insignificant when capacities are observed. For *Hill*, the trade-off with computational effort is not favourable, especially in comparison with *PET*. The *PET-Hill* approach does produce the best feasibility rates (as expected) in all cases but the improvement over *PET* is insignificant while requiring a little additional computational effort.

4.7 Conclusion

This chapter has demonstrated that a modified ANTCOL algorithm is capable of scheduling timetables on a first-order level. This chapter has investigated two scenarios.

Firstly, the *Non-Determined Timeslot Case*, in which the number of timeslots is minimised, was tackled. The basis of the ANTCOL algorithm was used along with techniques that enhanced solution quality. These ANTCOL variants were compared against some

benchmark methods. It was shown that the basic ANTCOL algorithm (*Basic)* outperformed the best-found single pass heuristic (heuristic C) and returned the optimal for seven data sets (out of twelve). Improvements to *Basic* were obtained through mixing construction heuristics (*Amalgam CD*), trail intensification (*Elitism*) and hill climbing (*Hill*). *Amalgam CD* observed optimal solutions for ten data sets, *Elitism* returned eight and *Hill*, eleven. These results compare very well in comparison to benchmark results.

Secondly, the *Determined Timeslot Case*, in which the number of timeslots is stipulated, was investigated. The graph was extended to accommodate timeslot vertices to accommodate additional constraints such as pre-assigned and time-windowed exams. A series of enhancement techniques were used and it was shown that the introduction of a trail that penalised difficult exams worked very well, which indicated the benefit of maximising the efficiency of the construction process rather than relying on repair facilities. When no timeslot capacities were observed, *Hill* was competitive (in comparison with *PET*) with respect to solution quality. However, the superiority of *PET* is evident when problems are constrained by seating capacities. In addition, *PET* based experiments require similar runtimes to *Basic* and significantly lower runtimes than *Hill*, thus increasing the attractiveness of *PET.* One disadvantage of *PET* resides with the efficiency of the algorithm itself. Success depends on the appropriate selection of parameter settings and construction heuristic since inefficient search environments will increase the number of *problem* exams and consequently, *PET* loses effectiveness (through differentiation).

PET was only applied to the *Determined Timeslot Case* in this chapter. However, scope does exist to utilise this diversification technique in the *Non-Determined Timeslot Case.* The user could specify the number of timeslots n and any exam allocated to a timeslot greater than n could be subject to trail replenishment. Alternatively, a simple greedy heuristic could be used to produce an upper bound n on the minimum number of timeslots. The ant algorithm can then be used in attempt to produce a solution in $n-1$, $n-2$ colours etc...until no solution can be found.

It is also demonstrated that standardising the reward function aids the search process. The fitness function used in Costa and Hertz (1997) is inappropriate and can be explained as follows: for two given data sets with solutions an identical distance away from their optimum, the rewards will be different if the optimum number of colours differ also.

Therefore, reduced differentiation between solutions exists for the some problems (data sets with bigger optimums). The reward function in Costa and Hertz (1997) is substituted with the function that quantifies the number of exams not allocated to the first *timeslot* colours.

This chapter shows that the ANTCOL algorithm with modifications is capable as an examination timetabling tool on a first-order level. To represent the application of ants to this problem area, the notation AS-EXAM is used forthwith.

Chapter 5

Minimising Second-Order Conflict

5.1 Introduction

Chapter 4 has proven that ants can successfully schedule examination timetables on a first-order level. In Chapter 5, we extend this methodology to accommodate second-order considerations. Second-order constraints are soft and aim to create a friendlier timetable. The definition of 'second-order' varies between institutions. For this investigation, we will use the criteria used at University of Wales Swansea and the second-order conflicts score of a timetable will be the number of occasions that students have to sit back-to-back exams. A student sitting two exams in adjacent timeslots crossing two exam days will be classified in the same way as two exams in adjacent timeslots on the same day. The objective is to minimise the second-order score of a timetable, thus giving the students as much revision time as possible.

Chapter 4 indicated that ants, when used collectively, were capable of producing large volumes of feasible timetables and it may be possible that a good solution with respect to second-order conflict may result. But, these would be produced by chance rather than by any pre-determined method. Therefore, this chapter aims to introduce methods to bias the ants towards areas of the solution space that contain feasible good quality timetables.

This chapter has the following structure. Firstly, it is demonstrated that the order in which the timeslots are constructed has an influence on the second-order efficiency of a timetable. Secondly, the second-order characteristics of a timetable are passed to the trail through the introduction of a modified reward function that quantifies both first and second-order attributes. Thirdly, a factor is introduced to encourage insertion of exams in timeslots that will result in lower second-order scores. Fourthly, the trail is intensified to enhance good solutions or more specifically good solution characteristics. Since exploitation techniques reduce the exploratory power of the ants, it is demonstrated that limiting trail levels improves the search conditions for the ants. Fifthly, the greater suitability of the DSatur heuristic for the examination timetabling problem is discussed but it is shown that the RLF heuristic is preferred due to its first-order superiority. Finally, the chapter refocuses with

the introduction of a second trail, which attempts to exploit good second-order characteristics.

It is worth noting the component settings of the basic method that are used in this chapter.

Component	Setting
α	2
β	1
ρ	0.5
Construction heuristic	C
N_A	100
N_C	100

Table 5.1 – Component Settings to be used unless otherwise stated.

The enhancement techniques that have been introduced in Chapter 4 do not feature here. It has been observed that the first-order performance of the basic algorithm is satisfactory (when we consider uncapacitated problems) and since the objective of this chapter is to improve second-order condition of timetables, it is felt justified, at this stage, to remove the first-order enhancements in order to place as much focus on the second-order problem as possible.

5.2 Second-Order Capabilities of First-Order Trail Model

The simplest way of reducing second-order conflict is to change the order in which timeslots are considered. Typically, filling them in order leads to high numbers of clashing exams in the initial periods and relatively empty late periods. In this section, we will contrast the results of filling timeslots in order with the results of filling timeslots in other orders.

In this section, second-order statistics are recorded when using the *Uncapacitated Non-Determined Timeslot Case* (Section 4.3) and do not accommodate any techniques to enhance second-order quality. These results will act as comparison for second-order enhancement strategies that are introduced later in this chapter. Strategies titled *Ordered*, *Random* and *OddEven* investigate different methods of selecting which timeslot to construct and illustrate the influence upon second-order quality of a timetable. These strategies are defined as follows. *Ordered* takes each timeslot in ascending sequence and allocates exams appropriately. With respect to second-order, the main disadvantage lies

with the exams to timeslots distribution. The majority of the exams are scheduled during the early phases of the timetable and will consequently inflate the second-order scores. Meanwhile, *Random* randomly selects which timeslot to construct next. The reduction in second-order conflicts is axiomatic and can be related to a more equal distribution of exams throughout the timetable. The third strategy, *OddEven,* fills the timeslots to create an uneven exam to timeslot distribution. The odd numbered timeslots are filled first and then the even numbered timeslots are dealt with. The selection of each timeslot is performed randomly within the *Odd* or *Even* group. This construction strategy encourages lower back-to-back conflicts due to the lower number of exams that are allocated to the even numbered timeslots. Results associated with these three ordering philosophies are as in Table 5.2.

	HEC			EAR			TRENT		
Strategy	*Av*	*Best*	*Feasible*	*Av*	*Best*	*Feasible*	*Av*	*Best*	*Feasible*
Ordered	2344.05	1458	87.08	2512.71	1953	62.71	2350.40	1923	79.86
Random	1972.82	939	87.39	2221.78	1572	72.78	1977.74	1491	85.27
OddEven	1646.40	926	86.15	1745.50	1430	73.38	1649.83	1255	81.07

Table 5.2 – Second-Order Statistics for Ordered, Random and OddEven Strategies.

Av represents the average second-order score over all feasible timetables, *Best* refers to the best 'one-off' observation and *Feasible* quantifies the average number of feasible timetables constructed per cycle. These statistics will be used forthwith.

Table 5.2 indicates the benefit that can be gained by selecting the appropriate method, which decides the timeslot to construct next. It is the framework *OddEven* that produces constructions that contain the lower second-order conflict scores and will be used forthwith. Any differences in the feasibility rates can be attributed to the randomness in the experiments.

Figures 5.1-5.3 present the average number of students per timeslot that are due to sit exams in adjacent timeslots for each of the proposed timeslot ordering strategies.

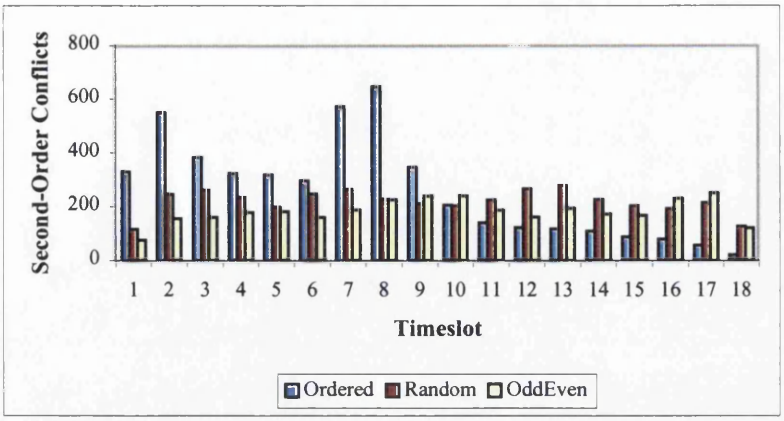


Figure 5.1-Average Second-Order conflicts per timeslot across construction methods for HEC.

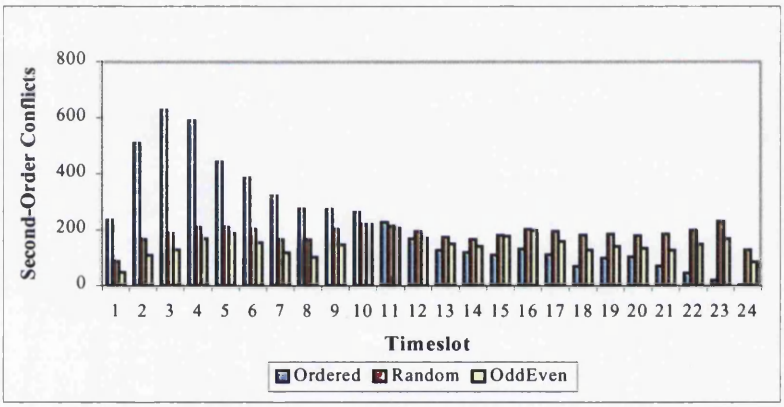


Figure 5.2 - Average Second-Order conflicts per timeslot across construction methods for EAR.

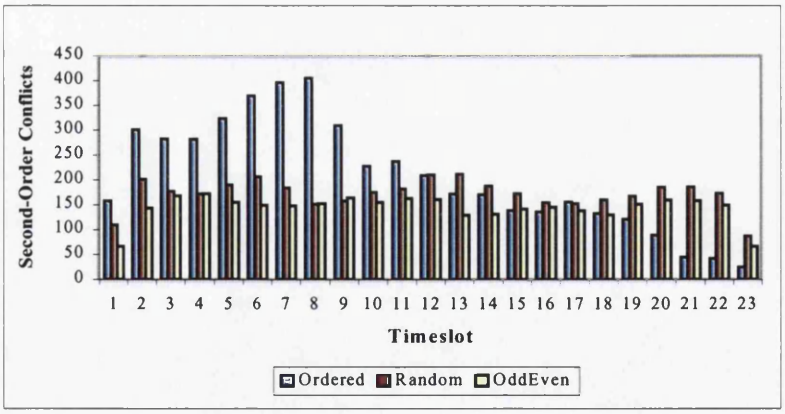


Figure 5.3 - Average Second-Order conflicts per timeslot across construction methods for TRENT.

The three construction procedures produce varying distributions of students to timeslots, which can be traced to the order that the timeslots are filled. *Ordered* exhibits a skewed distribution that inflates back-to-back conflicts during the first *timeslots/2* timeslots. As

expected the volume of students tails off towards the latter phases of the timetables due to the lack of exams allocated to those timeslots. However, the second-order conflicts experienced during the infancy of the timetable are costly and contribute to relatively high overall second-order conflicts scores. *Random* is a more suitable ordering technique since it encourages variation and consequently, the exam to timeslot distribution is more even and reduces the volume of students experiencing back-to-back exams as compared to the *Ordered* method. When using *OddEven*, the selection of the next timeslot to schedule is chosen in a pseudo-random manner. As detailed above, the odd numbered timeslots are placed in the first group of slots, with the even numbered timeslots allocated to the second group. When the first group becomes an empty set we move onto the second group. The probability of selecting a timeslot within these groups is purely random. This approach attempts to reduce the second-order scores, by design, through allocating higher numbers of exams to the odd timeslots and also, incorporates the *Random* method to encourage partial randomness. Work in Chapter 3 stressed the importance of variation.

Figures 5.4-5.6 represents the average second-order quality among feasible timetables over cycles for each of the timeslot selection strategies. As indicated by the statistics in Table 5.2, the performance of the selection strategies is consistent across the data sets with *OddEven* returning the more favourable results, followed by *Random* and then *Ordered.* We also note the lack of second-order improvement during the process, which indicates the need to incorporate second-order characteristics. Additionally, it is noted that the search stagnates with *Ordered* at a very early stage.

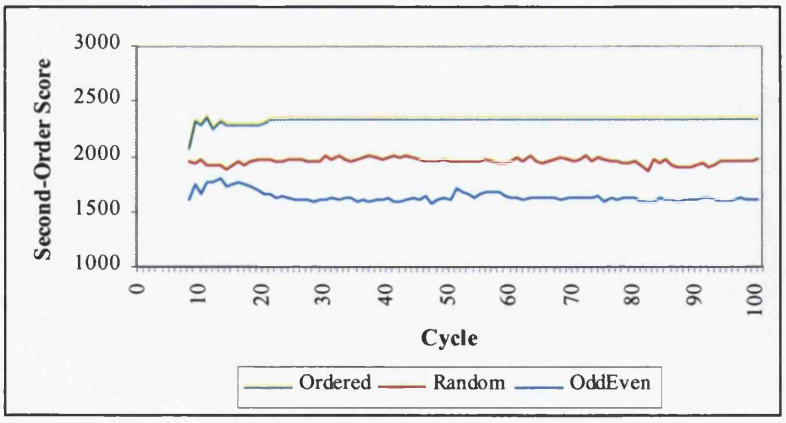


Figure 5.4 - Average Second-Order Score per Cycle across Construction Methods for HEC

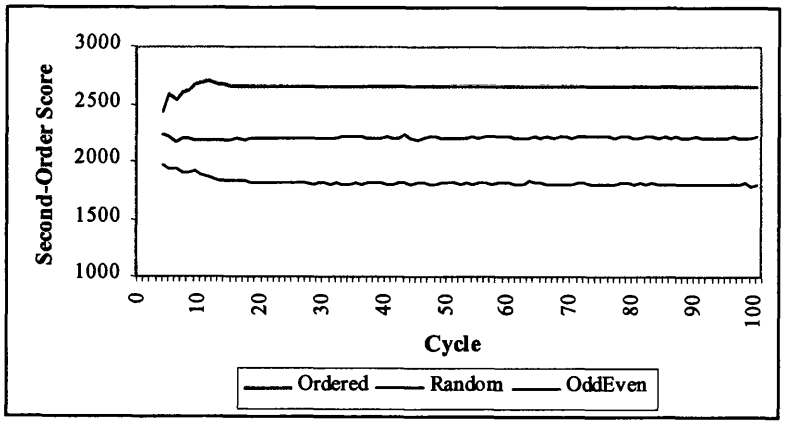


Figure 5.5 - Average Second-Order Score per Cycle across Construction Methods for EAR.

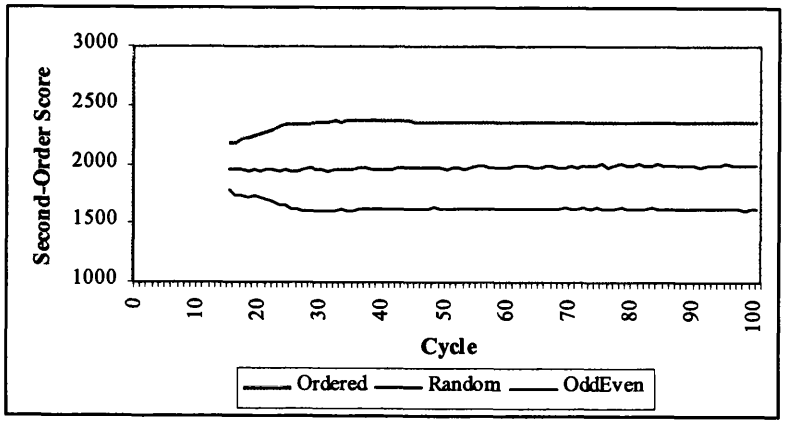


Figure 5.6 - Average Second-Order Score per Cycle across Construction Methods for TRENT.

The plots within Figures 5.4-5.6 start after the first cycle. Each starting point reflects when feasible timetables were first observed for all methods. This style of representation will be commonplace for the reminder of the thesis.

A possible variant of the *OddEven* ordering strategy is to construct the *Odd* timeslots and then allocate exams to exam blocks and allocate these exam blocks deterministically to the timeslots according to a greedy method that minimises second-order conflict. Like all greedy methods, large costs will be experienced towards the end of the procedure. If no second-order enhancement techniques are incorporated then such a variant is appropriate, however it will reduce the randomness within the search and future second-order enhancement methods presented will undoubtedly benefit from greater inherent variability (evidence detailing the influence of randomisation is presented in Chapter 3). Additionally,

this greedy technique would require the removal of timeslot vertices from the graph, which are required for the accommodation of pre-assigned and time-windowed exams.

Socha et al. (2002) discussed the importance of relative and absolute position of exams within a timetable. Please refer to Chapter 2 for a review of this paper. It can be seen that despite the importance of the relative position of exams with respect to second-order conflict, the knowledge of a preferred timeslot adds extra search potential. The inclusion of timeslot vertices within the graph allows the accommodation of both relative and absolute trail information. Therefore, not only the trails between pairwise exams within each timeslot are modified at each update stage but also the trail between the timeslot vertex and all exams belonging to that timeslot. Evidence of this additional benefit can be seen later in this Chapter (Section 5.7).

5.3 Combined reward function (CRF)

Chapter 4 discussed that the reward function used by Costa and Hertz (1997) could be improved through standardisation. It was suggested that the reward function should relate to the number of exams that could not be accommodated within the first t timeslots (where t equals the pre-specified number of timeslots or known-best solution). The reward function used is described in Equation 5.1.

$$\frac{1}{K+1} \quad (5.1)$$

where K equals the number of unallocated exams. Equation 5.1 was used in the *Determined Timeslot Case*, which constructs up to *timeslot* groups and then inserts the unallocated exams into the timeslots that create the lowest direct clashes. This approach is important since infeasible timetables are allowed to contribute to the trail and estimates of the second-order scores of infeasible timetables are necessary. However, it was noted in Chapter 4 that infeasible timetables are penalised collectively through Equation 5.1 and no extra trail is placed on the unallocated exams. Consequently, *PET* was introduced as a diversification tactic to move the search away from solutions that fail to colour certain exams by weighting the trail of unscheduled exams. It was demonstrated that the rate of

feasibility did increase through *PET*, however it is proposed here that *PET* is not used at this juncture in order to place more emphasis on the second-order problem.

In the nature of AS, all ants contribute to the trail (whether their solutions is feasible or not) due to the better solution quality that is experienced. However, some authors, for example Burke et al. (1994a), suggest otherwise.

In this section, we extend Equation 5.1 and introduce a weighted linear function to describe the perceived goodness of a timetable based on the balance between first and second-order priorities. The weights applied represent the relative unsuitability of constraint violation. For example, it is imperative that feasibility is observed and therefore, the penalty associated with first-order violations will be of the highest value. The reward function becomes the CRF and is as follows

$$\frac{1}{\mathrm{K}+1}+\frac{\partial}{Sec\_Score} \quad (5.2)$$

where, K refers to the number of exams not allocated to a timetable, *Sec_score* quantifies the second-order conflicts score of the timetable and δ is a selected constant. When $K>0$, the timetable is deemed as infeasible. Equation 5.2 is formed to represent a balance between first and second-order and conforms to a simple linear penalty-weighted sum as advised by Corne et al. (1994). However, some authors, such as Taillard (1993) and Di Gaspero and Schaerf (2000), disagree with the use of fixed weights and propose that dynamic weights can be used instead to adjust according to the nature of the problem and the structure of fitness landscape that the problem imposes. With respect to our study, it will be observed that feasibility is achieved relatively easily and consequently, greater attention is paid to minimising the second-order scores of timetables and therefore, it was felt unjustified to utilise dynamic weights at this stage.

The second-order component in Equation 5.2 is weighted by δ in order to find a balance between the first and second-order reward contributions. Too much weight places extra emphasis on the second-order characteristics of a timetable and dilutes the influence of the first-order information provided. Conversely, too little weight reduces Equation 5.2 to an

approximate first-order reward function and offers minute second-order communication. The second-order statistics for a range of δ are presented in Table 5.3.

	HEC			EAR			TRENT		
δ	Av	Best	Best 10	Av	Best	Best 10	Av	Best	Best 10
0	1601.45	942	1057.36	1773.62	1362	1491.20	1604.21	1268	1330.72
1	1748.81	895	1045.60	1744.91	1299	1454.88	1586.04	1247	1320.06
10	1745.09	910	1025.88	1748.50	1360	1495.68	1581.70	1235	1319.66
25	1649.92	890	1063.00	1751.02	1423	1504.60	1603.99	1235	1318.86
50	1687.64	916	1007.64	1791.28	1386	1537.34	1605.35	1208	1317.90
75	1507.56	933	1011.44	1780.80	1414	1495.50	1593.75	1195	1297.12
100	1667.44	912	1063.34	1648.71	1318	1452.14	1628.40	1289	1349.82
250	1684.85	894	1029.85	1761.68	1332	1505.84	1632.20	1234	1333.82
500	1501.12	868	1074.38	1793.56	1282	1505.86	1626.88	1222	1341.80
1000	1711.20	940	1061.18	1776.23	1297	1597.04	1633.63	1236	1324.38
2500	1703.98	841	1158.66	1753.03	1386	1502.46	1602.00	1211	1341.80
5000	1559.62	858	1156.64	1682.70	1323	1481.00	1607.36	1263	1342.48

Table 5.3 – Second-Order Statistics for Range of δ on the CRF

Best 10 refers to the average second-order conflicts score for the best 10 timetables per independent run.

Table 5.3 demonstrates that the influence of δ seems negligible, even for high δ. This could be attributed to the problem of conflicting information. The trail is trying to evaluate, for a pairwise set of vertices, how good the solution is if they are together, but this has no direct affect on how good the solution is in terms of second-order. Despite this, it seems intuitive that $\delta \geq 1$ since some representation of second-order should be present. The percentage of feasible timetables across the experimental range of δ is illustrated in Figure 5.7.

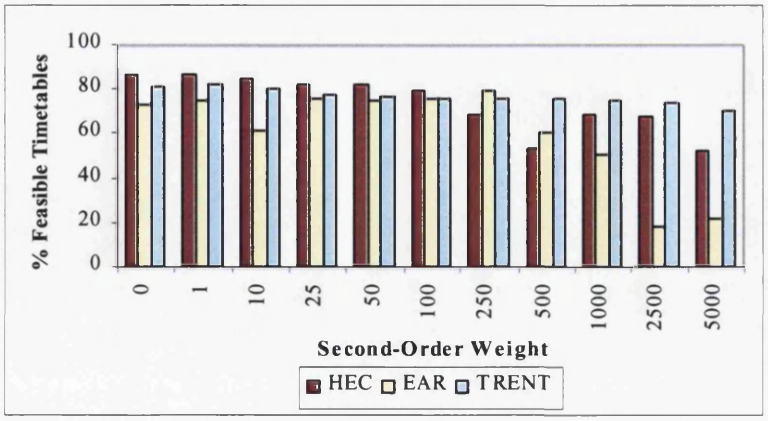


Figure 5.7 - Percentage of Feasible Timetables across Second-Order Weight δ.

The results in Table 5.2 contradict the writer's intuition regarding the influence of δ on second-order solution quality. It was anticipated that increasing δ would have a positive effect on the second-order scores. However, the results appear rather insensitive to the choice of δ. In addition, there is a negative impact on feasibility as δ increases. These two observations would suggest that there is no benefit in incorporating any second-order information within the fitness function. However, future enhancement techniques will demonstrate that improvement, regarding second-order performance, is achievable and consequently, some degree of second-order representation (see above also) is required. An arbitrary measure of $\delta=100$ is selected here. Subsequent studies will consequently calibrate other parameter settings accordingly.

Differences in the raw second-order scores (across all data sets) do exist and stating a universal value for δ across all data sets could be misleading and a precursory investigation that examines feasibility rates for various δ is suggested. However, all experiments performed in the remainder of this thesis will set $\delta=100$ and additionally, this can be used as an initial value if any fine-tuning is performed when obtaining δ for alternative data sets.

.

5.4 Second-Order Bias Term

The *RPR* is extended to accommodate a bias term, $[\eta(b[k-1],e,t)]$ (where *b[k-1]* relates to the partial solution at the k^{th} stage), that influences the ants to choose exams e that are favourable for insertion into timeslot t with respect to second-order. There are two definitions for this term and they vary according to the stage of the construction. When the odd numbered timeslots are being filled, $[\eta(b[k-1],e,t)]$ = 1 since there are no exam members in the even numbered timeslots. When the even numbered timeslots are regarded, $[\eta(b[k-1],e,t)]$ is interpreted as the inverse of the total clashes involving an exam e available for insertion in timeslot t and all exams in adjacent timeslots *t-1* and *t+1* while weighted by some constant θ. Let T be the set of unallocated exams that can still be inserted into timeslot t. Formally, the probability (p_{et}) of allocating exam e to timeslot t now becomes

$$p_{et} = \frac{[\tau(b[k-1],e,t)]^{\alpha} \cdot [\nu(b[k-1],e,t)]^{\beta} \cdot [\eta(b[k-1],e,t)]}{\sum_{r \in T} \{[\tau(b[k-1],e,r)]^{\alpha} \cdot [\nu(b[k-1],e,r)]^{\beta} \cdot [\eta(b[k-1],e,r)]\}} \quad (5.3)$$

When constructing timeslot t, there will be set of exams allocated to timeslot t-1 and a set of exams inserted in timeslot t+1. A matrix C is used, which refers to the number of students in common for each pairwise exam set. We have

$$\eta(b[k-1],e,t) = \begin{cases} 1 \text{ for odd timeslots} \\ \dfrac{\theta}{\sum_{m \in (t-1)} C(m,e) + \sum_{n \in (t+1)} C(n,e) + 1} \text{ otherwise} \end{cases} \quad (5.4)$$

A range of θ is examined and the associated results are presented in Table 5.4.

	HEC			EAR			TRENT		
θ	*Av*	*Best*	*Best 10*	*Av*	*Best*	*Best 10*	*Av*	*Best*	*Best 10*
0	1684.85	894	1029.85	1648.71	1318	1452.14	1628.84	1289	1349.82
1	1493.80	864	1029.06	1757.13	1420	1501.68	1513.87	1167	1257.92
10	1571.83	887	962.86	**1610.44**	**1299**	1423.93	1522.23	1161	1249.94
25	**1438.29**	**859**	914.54	1744.47	1426	1550.95	**1489.45**	1180	1239.60
50	1444.72	859	917.16	1691.32	1343	1483.05	1507.54	1140	1224.60
75	1444.96	859	918.30	1748.61	1343	1607.83	1520.21	1141	1242.46
100	1444.06	859	916.60	1712.64	1306	1525.30	1533.53	1121	1251.88
250	1442.99	859	916.04	1699.38	1317	1517.40	1547.78	1121	1268.48
500	1441.66	859	916.04	1695.85	1297	1517.58	1548.04	1125	1270.56
1000	1440.30	859	914.72	1705.44	1303	1530.63	1552.31	1224	1270.88
2500	1440.97	859	915.58	1751.22	1304	1568.30	1544.92	1114	1264.38
5000	1442.09	859	916.24	1696.98	1304	1436.15	1548.91	**1110**	1271.40

Table 5.4 –Second-Order Statistics for range of δ on Bias Term.

When θ=0, the bias constant is redundant and the related statistics will act as a comparison for alternative experimental values of θ. The inclusion of bias does improve solution quality. However, this claim is tenuous with respect to the EAR data set. With respect to HEC, setting $\theta \geq 25$ produces *Av* solutions that are approximately 200 lower than the no bias system and there is a slight improvement of the *Best* solution. As θ increases, the improvement in solution quality stagnates. With respect to TRENT, improvement of *Av* is

observed but not as significantly as for HEC. Setting $\theta \geq 1$ encourages *Av* qualities that are approximately 100 lower than $\theta=0$. However, the improvement of the *Best* solution is very encouraging, with the exception of $\theta=1000$. Each best *Best* and best *Av*, for the data sets, have been bolded.

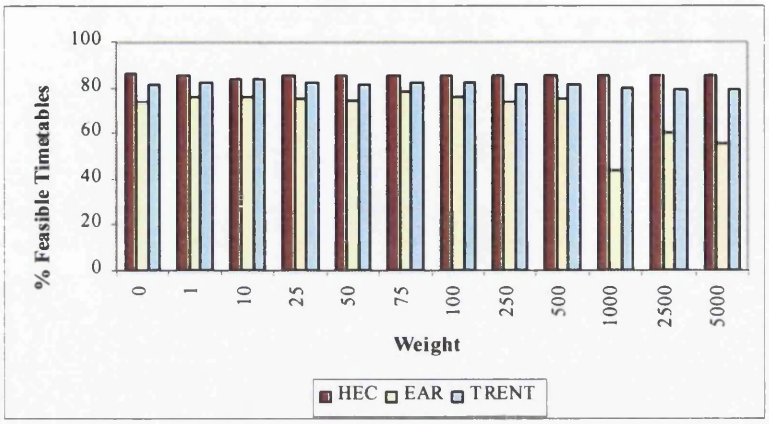


Figure 5.8- Percentage of Feasible Timetables across Bias Weight θ

Figure 5.8 presents the volume of feasible timetables for each of the main data sets across bias weight θ. As θ increases, there is a minimal drop in the feasible timetable rate with HEC and TRENT. This small reduction in first-order quality is profitable due to the average improvement in the second-order quality of the timetables. However, with EAR, there is a grave drop in first-order solution quality when θ gets large and no real compensation is evident with respect to second-order improvement. It is felt justified to use $\theta=100$ in future experiments since this setting improves second-order quality, maintains first-order feasibility and allows the search emphasis to remain with the ants rather than a deterministic greedy rule.

5.5 Trail intensification

Until this stage, trail replenishments have been linear and based upon the *CRF*. This section discusses methods of further differentiating between solutions based upon their perceived level of quality. The concepts discussed here have been inspired through either basic intuition or methods published in literature. Firstly, a static approach to additional trail replenishment is introduced. The trails associated with the ants that construct timetables with second-order conflict scores below the upper bound are intensified by

some predetermined value χ. Secondly, the natural progression to a dynamic approach is made. The upper bound is sensitive to the conditions of the run. Thirdly, a Modified Elitist Strategy, e.g. Dorigo et al. (1991), is incorporated within the algorithm. During a run, some solutions are regarded as favourable and the trails associated with these are intensified according to the weighting parameters used.

5.5.1 Static approach

The *CRF* is a linear function and consequently, the marginal increase in reward from one unit of fitness to the next is identical. This section introduces a basic method of placing greater reward on a subset of solutions. Components of the 'superior solutions' will have a greater probability of involvement in future constructions due to the increased trail levels.

In this section, we will employ a static approach that places extra reward upon solutions that have second-order scores below a predetermined upper bound (*UB*). The new reward function is defined mathematically as follows

$$\frac{1}{\mathrm{K}+1}+\frac{\partial}{Sec\_score}\times\chi \quad (5.5)$$

where

$$\chi=\begin{cases}\lambda & if\ Sec\_score \le UB\\ 1 & otherwise\end{cases} \quad (5.6)$$

and λ and *UB* are constants that are varied. *UB* was chosen according to the general range of solutions returned for each of the data sets. All ants are eligible for trail intensification.

The results for various *UB* and λ are presented in Appendix 5.1 but a summary of averaged statistics is presented in Table 5.5. Both *UB* and λ are divided into broad groups. For all data sets, the following intensity levels (λ) apply: *Low* - λ=5, 10 and 15, *Medium* - λ=20, 25, 30 and 35 and *High* λ=40, 45 and 50. The upper bounds for HEC are as follows: *Low*

– *UB*=1000, *Medium* - *UB*=1250, *High* - *UB*=1500 and *Very High* - *UB*=1750. For EAR and TRENT, the bounds are: *Low* – *UB*=1250, *Medium* – *UB*=1500, *High* – *UB*=1750 and *Very High* – *UB*=2000. Results are also compared again *None* (λ=1).

		Upper Bound Level			
	Intensity Level	*Low*	*Medium*	*High*	*Very High*
HEC	*None*	**1571.32**			
	Low	1477.05	1249.37	1299.89	1320.24
	Medium	1234.99	1236.45	1245.49	1320.51
	High	1197.49	1184.01	1204.48	1304.22
EAR	*None*	**1650.19**			
	Low	1650.19	1642.62	1585.95	1585.30
	Medium	1650.19	1570.87	1560.00	1627.31
	High	1650.19	1532.79	1392.56	1604.09
TRENT	*None*	**1397.87**			
	Low	1396.98	1414.44	1424.12	1493.73
	Medium	1371.26	1379.32	1435.89	1473.97
	High	1372.05	1317.30	1421.43	1491.98

Table 5.5 –Averaged Second-Order Scores for various Intensity and Upper Bound Levels.

The results indicate that some form of trail intensification is advantageous. The *Av* solutions do lower for HEC and EAR across all settings of *UB* as λ increases. Better 'one-off' solutions are also possible (see Appendix 5.1). With respect to all data sets, poorer solution quality is obtained through *Very High* in comparison to *Medium* and *High* and occasionally *Low*. This demonstrates the (potential) benefit of placing extra emphasis on a small subset of solutions.

The feasibility rates do drop as λ increases due to the higher reward of good second-order attributes and the reduced emphasis placed on the first-order condition of the timetable. The *UB* does also have a controlling role on feasibility. A higher *UB* allows for a greater number of eligible solutions that are intensified by λ. Consequently, the relative differences between solution qualities are closer to the non-intensified system than desired. This leads to limited improvement (with respect to second-order) but the maintenance of feasibility.

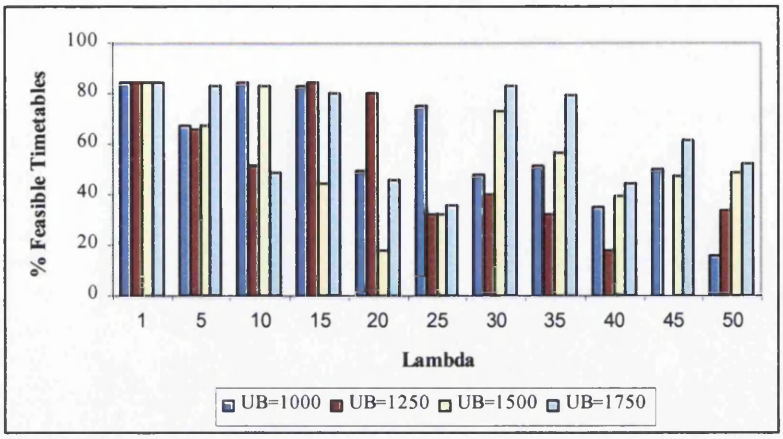


Figure 5.9 - Percentage of Feasible Timetables across Intensification Weight λ for HEC

Figure 5.9 provides an illustration of the impact of λ and *UB* on feasibility. It can be seen that feasibility deteriorates as λ increases. The influence of *UB* is interesting in this instance. It can be seen that higher feasibility is maintained for the largest value of *UB* (i.e. *UB=1750*) for higher λ, which corresponds to what has been stated above.

This section has illustrated the influence of placing extra reward on a subset of solutions. The second-order improvement is not as great as desired but some solution enhancement has been shown. The biggest lesson is the impact on feasibility, which demonstrates the need to be cautious in future studies that offer additional replenishment for trail associated with good second-order components.

5.5.2 Dynamic approach

The drawback of Section 5.5.1 resides with its static nature. Before such a method can be employed, insight into the typical range of second-order conflict scores for each data set is needed. Only then can the upper bounds be determined and the static approach performed. The implementation of a dynamic approach removes the need for any pre-investigation runs, but does require the definition of some acceptance criterion – a method that defines the upper bound. The approach in this section has been inspired by the work detailed in the Record-to-Record Travel (RRT), Dueck (1992), which derives from the Great Deluge Algorithm (GDA), which was presented in the same paper as RRT. Both algorithms are one parameter optimization heuristics that set acceptance rules for worse intermediate solutions while exploring a solution space. The author proposed the GDA and the RRT

algorithms to provide alternatives to Threshold Accepting (TA) and Simulated Annealing (SA).

Pseudo-code for the RRT algorithm is as follows, as from Dueck (1992). This is a maximisation problem and will be edited for minimisation.

Choose an initial configuration
Choose an allowed DEVIATION>0
Set RECORD=quality of initial configuration
    DO
    Choose a new configuration, which is a stochastic small perturbation of the old configuration
    Compute E=quality (new configuration)
    If E>RECORD-DEVIATION
        THEN old configuration=new configuration
    If E>RECORD
        THEN RECORD=E
    UNTIL there has been no improvement for a predetermined number of iterations

RRT, like GDA, requires the selection of one parameter for good performance compared to a sequence of parameters like some methods. The RRT provides the basis to the work in this section: a dynamic upper bound approach to trail intensification. Here, the algorithm is applied to minimization of costs (second-order) rather than any maximization problem. Firstly, the upper bound that is used to decide eligibility for trail intensification is some measure (deviation) away from the run-based minimum. The trail associated with any solution that has second-order costs below the upper bound is intensified by some factor θ. When a feasible solution is lower than the run-based minimum then the minimum is reset as the newfound run based minimum. The pseudocode for this approach is as follows:

Choose an allowed percentage deviation ε
Choose a replenishment rate θ
Intensify First Solution and set Best Solution=First Solution
    DO

Set Upper Bound=Best Solution + ε*Best Solution

Generate next solution and calculate quality E

If E<Upper Bound Then

$$\text{Reward} = \frac{1}{\text{K}+1} + \frac{\partial}{Sec\_score} \times \theta$$

Else

$$\text{Reward} = \frac{1}{\text{K}+1} + \frac{\partial}{Sec\_Score}$$

Endif

If E<Best Solution Then

Best Solution=E

Endif

UNTIL all predetermined number of ants have constructed timetables

Results across ε are as follows in Table 5.6. The row in italics refers to the system described in section 5.4 when δ=100.

	HEC			EAR			TRENT		
ε	*Av*	*Best*	*Best 10*	*Av*	*Best*	*Best 10*	*Av*	*Best*	*Best 10*
-	*1444.06*	*859*	*916.60*	*1712.64*	*1306*	*1525.30*	*1537.53*	*1121*	*1251.88*
0.05	1581.02	752	995.12	1749.87	1326	1559.43	1451.65	1108	1182.68
0.10	1593.69	845	1024.76	1478.77	1175	1289.20	1423.48	1106	1181.44
0.15	1381.17	788	903.76	1472.19	1261	1311.10	1345.59	1071	1134.10
0.20	1464.79	925	972.20	1626.75	1278	1480.40	1412.95	1149	1202.36
0.25	1404.14	841	961.88	1640.05	1427	1488.50	1385.44	1101	1180.24
0.30	1508.70	721	1100.44	1672.04	1356	1484.80	1440.84	1044	1204.18
0.35	1193.60	803	897.62	1587.82	1256	1382.68	1440.54	1102	1211.96
0.40	1271.90	762	896.38	1603.67	1304	1474.27	1519.98	1135	1266.70
0.45	1294.73	847	1012.28	1462.43	1464	1548.00	1479.43	1142	1228.00
0.50	1179.23	814	886.92	1593.45	1282	1440.40	1511.20	1157	1268.44

Table 5.6– Second-Order Statistics for a range of ε

Generally, the results associated with the dynamic method seem similar to those achieved with static intensification. A potential downside of this approach lies with the use of a run-minimum. If a 'freak' good quality second-order quality timetable is produced early in the search no real benefit is gained from this approach since a limited number of ants intensify their trails. Thus, here we propose the use of a local-minimum rather than the run-minimum. The minimum score from the previous 5, 10 and 20 cycles is used. Before the 5^{th}, 10^{th} and 20^{th} cycles the run-minimum is used. With respect to the pseudocode

presented above the only alteration belongs to the *Best* Solution. While the *Best* solution found naturally remains the Best Solution, for the purposes of localising the minimum used, *Best* solution becomes *Best* solution$_{c\text{-}n}$, which is defined as the *Best* solution found during the previous *n* cycles, when in cycle *c*. For these experiments the ε values used are 0.25 for HEC, 0.10 for EAR and 0.15 for TRENT. These seemed sensible for the first and second-order quality trade-off.

	HEC			EAR			TRENT		
n	*Av*	*Best*	*Best 10*	*Av*	*Best*	*Best 10*	*Av*	*Best*	*Best 10*
-	*1404.14*	*841*	*961.88*	*1478.77*	*1175*	*1289.20*	*1345.59*	*1071*	*1134.10*
5	1324.60	720	907.70	1651.54	1317	1428.85	1348.54	1064	1156.80
10	1345.73	778	941.86	1536.97	1334	1413.85	1361.98	1028	1121.20
20	1366.64	837	962.58	1476.09	1180	1286.50	1333.47	1018	1119.22

Table 5.7– Second-Order Statistics for a sample of n

The minimums for both run and localised conditions are illustrated in Figures 5.10-5.12 for HEC, EAR and TRENT. These diagrams provide reasoning behind the results displayed in Table 5.7 There is an increase in solution quality with HEC through localising the minimum but not with EAR or TRENT. There is a notable disparity between the run and localised minimums used with HEC (see Figure 5.10) but not with EAR or TRENT. Therefore, similar volumes of ants intensify solution trails with EAR and TRENT across the types of minimums used but, with HEC, fewer solutions are intensified with the use of the run-minimum. Figures 5.10-5.12 indicate that not a large amount of ant communication is required in order to find the overall *Best* since it is found during the early stages of the search.

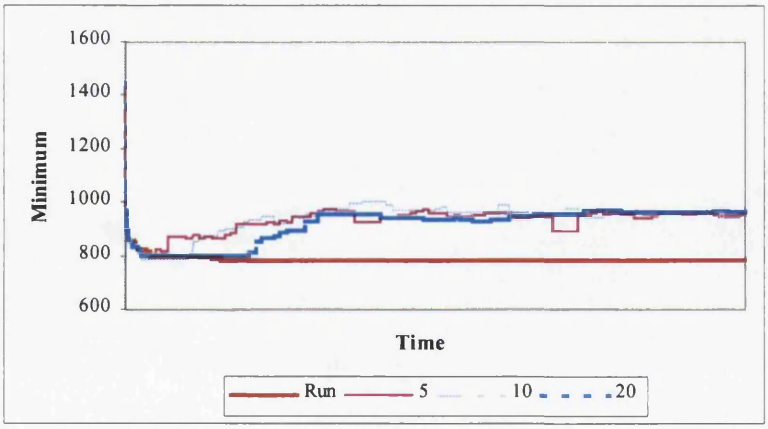


Figure 5.10 – Run, 5, 10 and 20 Cycle Minimums for HEC

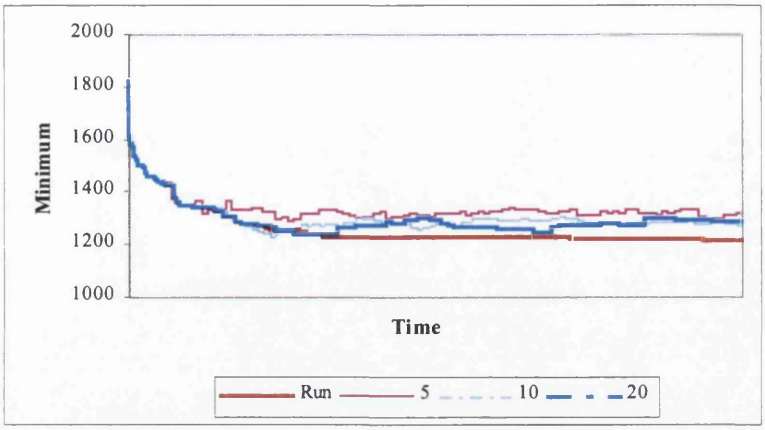


Figure 5.11 – Run, 5, 10 and 20 Cycle Minimums for EAR

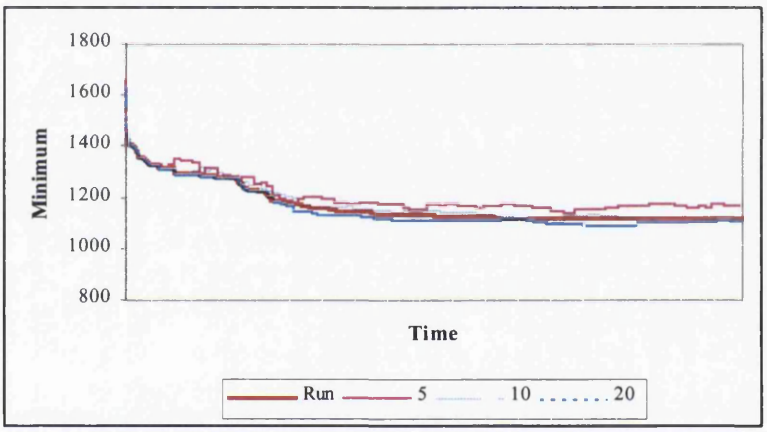


Figure 5.12 – Run, 5, 10 and 20 Cycle Minimums for TRENT

5.5.3 Elitist Ants

The use of elitism in evolutionary combinatorial methods was first suggested in De Jong (1975) and was then first applied to AS through the TSP problem by Dorigo et al. (1991). Elitism was touched upon in Chapter 4 when applied to the first-order problem. Its benefit was overshadowed by alternative solution enhancement techniques but evidence of improvement was presented. Additionally, Sections 5.5.1 and 5.5.2 demonstrated that benefit could be derived from intensifying top solutions. However, the decision to intensify was based on criteria that offered drawbacks. So, here we apply a method that intensifies a top percentage of solutions. Work in this section will demonstrate the additional exploiting

power that elitism offers and how crucial this method is with respect to lowering the second-order scores of a timetable.

When exploring a solution space, an elitist strategy places additional reward on solutions that are deemed 'elite'. Typically, an 'elite' solution refers to the best solution according to some criteria. For example, Dorigo et al. (1991) reinforced the edges on the best solution with additional trail quantity $e.\frac{Q}{L^*}$, where e is the number of elitist ants, Q is some predetermined constant and L^* is the length of the best tour constructed.

Preliminary experiments observed that applying elitism as in Dorigo et al (1991) and Dorigo et al. (1996) does not generate the results as desired. Therefore, the true form of elitism will not be applied here. Instead of intensifying the best solution e times, the top global e solutions will be considered and weighted by a factor σ. This allows the majority of the information to be passed on by the global fittest members of the population and relates to tournament selection technique in Genetic Algorithms where only the n fittest members produce offspring, Goldberg (1989). Non-elitist ants are still permitted to contribute in order to encourage diversity of search.

With respect to the examination timetabling problem, depositing additional pheromone on pairwise exam sets belonging to the *Best* solutions will potentially bias future ants to insert these pairwise exam sets into the same timeslot. The underlying aim of this approach is that the combination of this bias and the stochastic nature of the algorithm will lead to the construction of the optimum or near optimum solutions.

The elitist approach used here will attempt to reduce the amount of experienced second-order conflicts. This section will take the following structure. Firstly, the criterion for elitism is based on second-order scores only. Secondly, the definition for elitism is altered to accommodate the first-order aspects of the problem. The criterion for inclusion within the 'elite' list is consequently based on the Elitist Reward Function (ERF). Thirdly, and finally, this approach is extended to include additional second-order bias.

5.5.3.1 Second-Order Elitist Strategy

At each trail update stage per run (at the end of each cycle), the elite list is revised to contain ant members that have constructed timetables with the lowest second-order conflict scores. At each update stage, the trails associated with the *e* elitist ants are reinforced, while the level of trail between pairwise exam set (*i,j*) during cycle *t+1* can be computed as;

$$\tau_{ij}(t+1) = \rho.\tau_{ij}(t) + \Delta\tau_{ij} + \Delta\tau_{ij}^{*} \quad (5.7)$$

The level of trail at cycle *t* is subjected to an evaporation rate ρ and a replenishment update $\Delta\tau_{ij}$ (as before). The set of *e* elite ants passes additional information to τ via $\Delta\tau_{ij}^{*}$, which can be defined as;

$$\Delta\tau_{ij}^{*} = \begin{cases} \sigma.\dfrac{100}{Sec\_score} & \text{where } \sigma \geq 1 \quad (5.8) \\ 0 \ otherwise \end{cases}$$

σ represents the quantity of additional trail replenishment and *Sec_score* relates to the second-order score of a timetable. A constant *100* has been chosen based on the second-order component of the CRF used in previous experiments and has been held static for ease of comparison with the results derived from previous investigations.

To implement this strategy, some parameters need to be deduced - the unknowns that have been mentioned above. Firstly, the extra emphasis that is placed on the trail (σ) and secondly, the number of elitist ants used (*e*) per run. To achieve suitable estimates for these parameters some precursory information is necessary.

Only one parameter is varied at any one time while the other parameter remains as stated below. These original settings have been selected based on previous experiments and literature. Dorigo et al. (1991) indicated that a range of *e* could be chosen and return similar results. Meanwhile, White et al. (2003) used $(1/6)^{th}$ of the amount of cities in the

TSP. In this section, we will use 10% elitist ants and allow adjustment if necessary. Meanwhile, Section 5.5.1 revealed that an intensification weight of 15 is a good compromise between first and second order conditions and so should be regarded as a sensible starting point for these experiments.

Parameter	Setting
e	10
σ	15

Table 5.8– Experimental settings of e and σ

Influence of σ

We now vary the trail replenishment rate σ and the results are presented in Table 5.9.

	HEC			EAR			TRENT		
σ	*Av*	*Best*	*Best 10*	*Av*	*Best*	*Best 10*	*Av*	*Best*	*Best 10*
1	*1428.13*	*830*	*925.12*	*1678.23*	*1367*	*1472.39*	*1507.29*	*1174*	*1291.78*
5	1618.53	823	973.56	1640.85	1341	1447.73	1490.10	1122	1227.98
10	1574.17	829	934.02	1566.70	1332	1416.68	1444.22	1107	1172.30
15	1377.89	649	861.78	1556.22	1257	1402.50	1383.68	1087	1142.18
20	1246.18	814	893.88	1325.47	1155	1162.40	1373.52	1055	1142.22
25	1292.24	684	955.28	No feasible solutions			1397.19	1046	1176.86
30	1320.66	747	935.50				1316.34	1119	1171.64
35	1067.88	740	964.85				1336.79	1010	1157.73
40	1145.80	692	767.73				1247.78	1049	1099.02
45	1036.98	690	870.33				1365.48	1065	1209.43
50	1393.44	734	1100.25				1245.18	1015	1101.83

Table 5.9– Second-Order Statistics describing influence of σ

There is an encouraging improvement in the second-order conflict scores of the feasible timetables constructed as σ is increased. Overall, the average scores decrease and there is a noticeable improvement in the *Best* solution. However, as is commonplace with timetables of 'better' second-order scores, there is a marked decrease in the volume of feasible timetables generated. For $\sigma \geq 25$, no feasible timetables were recorded for EAR and the volumes associated with both HEC and TRENT are also reduced (see Figure 5.13).

As σ increases, non-elitist feedback becomes less influential. In consequence, the AS-EXAM system moves closer to the ACS and MMAS frameworks that only use the global best solution to update trails. The use of lower σ in AS-EXAM allows non-elitist solutions to have a greater role and thus, encourages greater diversity and effectively more exploration.

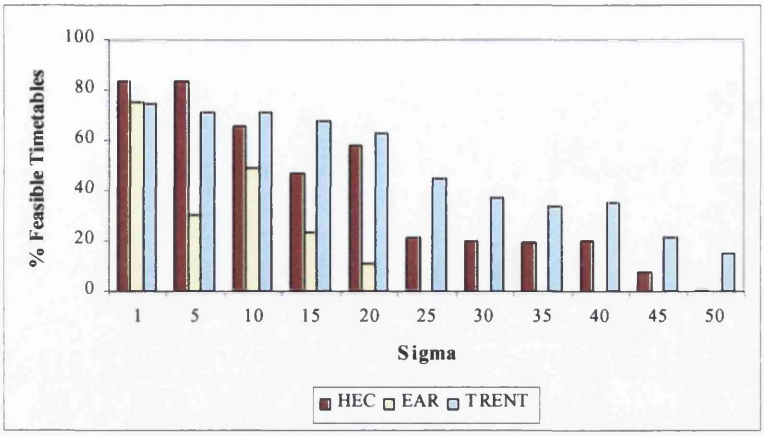


Figure 5.13 – Percentage of Feasible Timetables for range of σ for Second-Order Elitism

There is no exact approach to the deduction of appropriate parameter settings and there is no exception here.

A summary of results for each data set is a follows.

HEC

For $\sigma \geq 15$, feasible timetables with superior second-order scores are constructed. The *Best* scores are as low as *649*, which is very good. The trade-off with feasible timetable count is quite severe. For $\sigma \geq 25$, the percentage drops considerably. Initially the proportion drops to approximately *20%* and decrease below a *1%* average when $\sigma = 50$. However, the improvement in second-order quality dampens as σ reaches higher intensities. Setting $\sigma =$ *[15,20]* here seems appropriate. Even though the *Best* score (*814*) associated with $\sigma = 20$ is not near the best score observed, encouragement comes from a satisfactory *Best 10* score (*893.88*) and a reasonable feasibility rate.

EAR

As stated above runs performed with $\sigma \geq 25$ are not suitable due to the inability of the method to generate one feasible timetable. For weight intensities below 25, the second-order statistics improve progressively as σ increases. In response, as stressed previously, the volume of feasible timetables decrease. The most satisfactory second-order results for

EAR are returned when $\sigma = 20$ but only *10.81%* of the timetables are deemed as feasible. However, despite this, σ will be set at 20 when using the EAR data set as we only need to produce one good solution.

TRENT

Increasing σ does reduce the volume of feasible timetables generated but the decrease is not as drastic as for the other data sets. Firstly, feasible timetables are constructed for all settings of σ. Additionally, for the most intense setting experimented ($\sigma = 50$), a relatively respectable *15.03%* of the timetables are feasible. With respect to second-order statistics, there is an improvement as σ is increased. However, the benefit slows for relatively small settings of σ. Consequently, with the first and second-order trade-off in mind, a range of settings $15 \leq \sigma \leq 25$ appear justifiable.

In general, setting $\sigma=20$ seems appropriate for all datasets, however there may be potential problems with the feasibility rates of difficult data sets. Thus, a value of 15 has been chosen here, and in practice, a weight could be selected which adapts as the search progresses according to the number of feasible timetables being produced.

Influence of e

We now investigate the sensitivity of varying the number of elitist ants (e) used per run. The trail intensification rate, σ, is set at 15 for all data sets. The results are tabulated here:

		Number of elitist ants (e)						
		1	*5*	*10*	*15*	*20*	*30*	*50*
HEC	*Average*	1333.44	1351.37	1377.89	1462.01	1383.91	1355.73	1277.22
	Best	845	727	649	697	762	795	691
	Best 10	933.03	973.34	861.78	956.15	933.26	930.34	847.88
	Worst 10	2239.07	2106.30	2408.30	2153.68	2286.00	2403.80	1924.55
EAR	*Average*	1768.43	1529.58	1556.22	1566.24	N/a	N/a	1305.98
	Best	1425	1265	1257	1315			1140
	Best 10	1535.23	1377.20	1402.50	1406.50			1681
	Worst 10	2057.13	1748.57	1778.13	1901.70			1598.40
TRENT	*Average*	1443.85	1423.71	1383.68	1441.84	1339.06	1335.48	1438.65
	Best	1141	1056	1087	1146	1055	1009	1075
	Best 10	1229.08	1192.60	1142.18	1226.06	1145.12	1120.88	1277.70
	Worst 10	1819.80	1794.94	1782.66	1815.36	1724.86	1717.98	1708.62

Table 5.10– Statistics describing influence of e

Generally, the second-order performance of the algorithm improves as the proportion of elitist ants increases. However, such a statement should be taken tentatively. The indifferences of the second-order statistics for various e are such that any $e \geq 5$ appears acceptable. This can be explained as follows. Since the criterion for elitism rests only with the second-order score of a timetable, the elite list contains timetable constructions that have good second-order quality, irrespective of feasibility status. Members of elite lists are solutions of similar second-order quality and also, not too dissimilar constitutions. After some search time has elapsed, the pheromone levels associated with certain (good second-order) solution characteristics play dominant roles and have a strong influence the decisions of subsequent agents. Ants become trapped within infeasible regions of the solution space due to the attempts to exploit (strong) feedback that has been passed to the trail by previous ants. This strong feedback originates from the elite ants, which have the objective to identify good second-order characteristics. The influence on feasibility rates for experimental e can be seen in Figure 5.14.

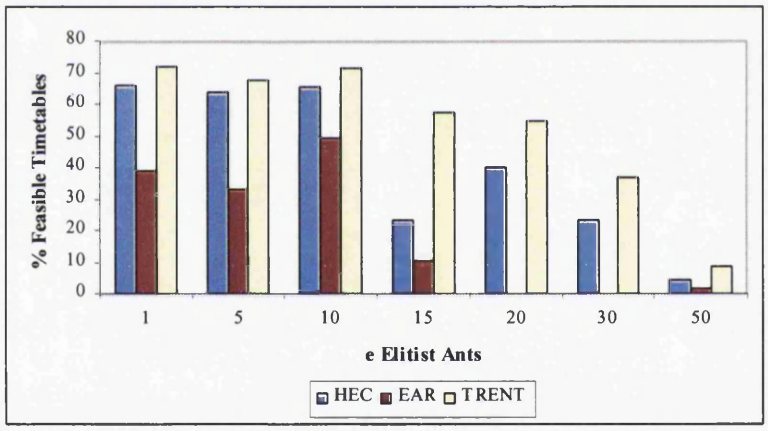


Figure 5.14 – Percentage of Feasible Timetables for range of e for Second-Order Elitism

Placing additional reward on a subset of trails that belong to 'better' timetables appears to improve the overall second-order solution quality of the timetables generated. Increasing the trail intensification weight, σ, has a positive influence on second-order aspects, while the impact of varying the number of elitist ants, e, holds less power. Magnifying both parameters has a detrimental effect on the first-order capabilities of the algorithm across all data sets. Observing the first-order statistics provides interesting insight. For example, with the EAR data set, when feasible timetables are difficult to come by, the average number of unallocated exams at mature phases of a run can be greater than 5 which, indicates attention needs to be paid to the first-order requirements of the problem. This would

require neglecting second-order aspects. Adaptive weights are an option, which slackens the focus placed on second-order importance at strategic times of the search i.e. at the observation of infeasibility. However, due to the search time limit (i.e.100 cycles), observing the benefit of sensitised elitist weights or the number of elitist ants is unlikely.

Observing the first-order problems during this phase of investigation became the springboard for the following piece of study: an elitist strategy that considers first and second-order characteristics. To complete this section, parameter settings that perform robustly are $\sigma=15$ and $e=10$.

5.5.3.2 Elitist Reward Function (ERF)

Section 5.5.3.1 concluded that an elitist strategy, with second-order bias, prompts the construction of timetables with second-order qualities better than previously experienced. While this was encouraging, the major flipside was the decrease in the feasibility rates that resulted. Within this section, we attempt to address the balance between first and second-order priorities. The membership of the elite list is no longer solely dependent on the second-order quality of the timetable but now it is based on a modified version of the CRF.

Similar to the previous section, the level of trail between pairwise exam set (i,j) during cycle $t+1$ can be defined as in Equation 5.9;

$$\tau_{ij}(t+1) = \rho.\tau_{ij}(t) + \Delta\tau_{ij} + \Delta\tau_{ij}^{*} \quad (5.9)$$

Whereas, $\Delta\tau_{ij}^{*}$ now can be defined as;

$$\Delta\tau_{ij}^{*} = \begin{cases} \sigma.\,ERF & \\ & \text{where } \sigma \geq 1 \\ 0 \;\; otherwise & \end{cases} \quad (5.10)$$

This additional replenishment depends on the ERF and trail intensification rate σ. ERF is defined as:

$$ERF = \frac{1}{\mathrm{K}+1} + \frac{100+\delta_2}{Sec\_score} \text{ where } \delta_2 \geq 0 \quad (5.11)$$

with σ representing the quantity of additional trail replenishment, *Sec_score* relating to the second-order score of an 'elite' timetable and δ_2 is the additional weight that raises the importance of the second-order condition of the timetable and is used for bias towards solutions of better quality.

During precursory investigation the parameters are set at the following;

Parameter	Setting
e	10
δ_2	0
σ	15

Table 5.11– Experimental settings of e, δ_2 and σ.

Influence of e

We now investigate the sensitivity of varying the number of elitist ants, e, used per run. The trail intensification rate, σ, is set at 15 for all data sets. The results are presented in Table 5.12.

		Number of elitist ants (e)						
		1	*5*	*10*	*15*	*20*	*30*	*50*
HEC	*Average*	1209.22	**1070.83**	1264.16	1225.31	1191.46	1187.57	1272.97
	Best	746	777	743	**703**	780	793	767
	Best 10	826.74	799.64	863.26	828.82	840.06	839.76	834.48
	Worst 10	2139.98	2274.54	2483.36	2437.18	2385.5	2551.46	2402.62
EAR	*Average*	1608.43	1548.28	1580.27	1455.76	**1453.9**	1557.06	1553.38
	Best	1240	1249	1208	**1088**	1148	1220	1212
	Best 10	1337.6	1284.2	1322.4	1196.6	1234	1281.4	1310.4
	Worst 10	2016.92	1987.92	2071.36	1945.02	1946.46	1993.14	1960.38
TRENT	*Average*	1404.05	1309.22	1405.28	1387.21	1322.6	**1301.47**	1435.68
	Best	1077	1013	1076	1034	**1011**	1021	1118
	Best 10	1163.86	1072.2	1152.64	1156.76	1078.5	1063.14	1177.76
	Worst 10	1825.32	1823.64	1882.34	1863.66	1869.46	1838.9	1897.9

Table 5.12– Second-Order Statistics describing influence of e

There is no distinctive pattern in the second-order statistics generated for various e. The *Average* and *Best* results for each of the data sets have been highlighted to emphasise the lack of pattern that exists amongst the results. With HEC, the most appealing average

second-order score is when $e=5$, while the *Best* result was generated for $e=15$. Consequently, no definite selection of e can be chosen when considering this data set. Similar statements can be made for the other data sets. Since the best solutions for HEC and EAR were produced when $e=15$ and a competitive best was found for TRENT at this setting we shall conclude that $e=15$ is arguably the more favourable of the settings of e. Encouragingly, the first-order capabilities are markedly better than experienced in Section 5.5.3.1.

As an aside, we should note that it is advisable to achieve a balance between exploration and exploitation. A small proportion of elitist ants will encourage stagnation since a small subset of solutions will dominate and bias future explorations strongly to form certain constructions. Conversely, a large number of elitist ants will limit the differentiation between types of solutions and the underlying aim of elitism will be lost.

Influence of δ_2

A weight δ_2 has been introduced to place additional importance on the second-order condition of a timetable. Previously, the weight δ in the CRF has been set at 100. However, by placing additional weight on the second-order component of the reward function, the elite list will become more susceptible to members dominated by their relatively good second-order scores. Consequently, less emphasis will lie with the first-order domain of a timetable. Increasing δ_2 has a positive influence on the overall second-order capabilities of the search. The *Av*, *Best 10* (statistics in Table 5.13 have been omitted) and *Worst 10* solutions decrease encouragingly as the second-order bias is intensified. The most notable improvement lies with the *Worst 10* solution score. Typically, with regard to the HEC data set, *Worst 10=2483.36* when $\delta_2=0$ and when δ_2 raises to 2900, for example, *Worst 10=1497.63*. This provides an indication of the drop in *Worst 10* scores when the weight of δ_2 is increased, showing the positive influence of increased measures of δ_2. However, it is not common to be concerned with the quality of the poorer solutions since we are only generally interested in the *Best* solution.

The results are presented in Table 5.13.

	HEC			EAR			TRENT		
δ_2+100	*Av*	*Best*	*Worst 10*	*Av*	*Best*	*Worst 10*	*Av*	*Best*	*Worst 10*
100	1264.16	743	2483.36	1580.27	1208	2071.36	1405.28	1076	1882.34
250	1221.60	656	2462.46	1469.30	1132	1983.20	1322.53	1007	1860.26
500	1280.52	725	2472.50	1429.86	1149	1907.78	1323.13	1003	1810.46
750	1121.60	754	2247.94	1328.36	1022	1777.35	1277.37	1019	1731.20
1000	1119.08	701	2330.28	1425.11	1161	1832.70	1302.75	1007	1794.06
1500	1029.31	676	2102.54	1279.98	1086	1599.15	1237.71	983	1722.70
2000	1111.25	732	2004.14	N/a	N/a	N/a	1303.83	1019	1773.38
2500	999.46	736	1719.55	1357.46	1177	1760.00	1248.83	999	1699.70
3000	908.01	732	1497.63	1337.01	1131	1717.40	1200.38	947	1690.04
3500	947.28	645	1535.73	1347.11	1247	1723.60	1233.81	949	1678.38
4000	1048.09	695	1990.20	1231.89	1111	1617.90	1251.04	1020	1709.20

Table 5.13– Second-Order Statistics describing influence of δ_2

The trade-off for improved second-order search capacity is the reduction in the volume of feasible timetables (see Figure 5.15). On occasions, increasing δ_2 reduces the ant's first-order ability to an extent that no feasible timetables can be constructed. In such circumstances, since no diversification criterion is used, the ants remain trapped within the infeasible areas of the solution space. As before, EAR proved to be the more difficult data set due to the number of infeasible runs for higher settings of δ_2 - when δ_2=*1900* (i.e. δ_2+100=2000), the algorithm did not generate one feasible timetable across five runs. For δ_2=2400, 2900, 3400 and higher, four infeasible runs are recorded. Infeasible runs are observed for the HEC data set, but to a lesser extent than EAR. However, no such problems are realised with experiments on TRENT. All runs across all predetermined settings of δ_2 return at least one feasible timetable.

Figure 5.15 illustrates the percentage of feasible timetables across δ_2. A downward trend is observed as δ_2 is increased. Feasibility rates drop considerably for higher δ_2 with HEC and EAR. Typically, for $\delta_2 \geq$*1000*, the feasibility rate for EAR is lower than *30%*, when δ_2=*2000*, no feasible timetables are constructed. Meanwhile, the percentage proportion of feasible timetables for TRENT does lower slightly as δ_2 is intensified, but the percentage remains fairly consistent. A robust setting of δ_2+*100* is *750* and will be used in future experiments, when appropriate.

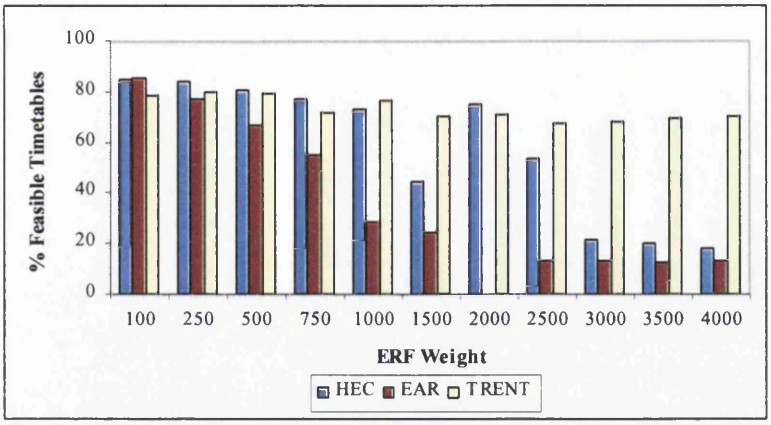


Figure 5.15 – Percentage of Feasible Timetables across ERF Weight δ_2+100

Influence of σ

Varying σ provides a way of further differentiating between solutions. The higher the weight, the larger the disparity in solution reward. Increasing the intensity improves the average performance of the search. The average second-order conflicts score and the average *Best 10* score drops fairly consistently as σ is magnified, although, interestingly, there is no improvement in the averaged *Worst 10* solutions. The *Worst 10* scores are stagnant across the experimental values of σ.

Increasing σ does not influence the first-order success of the algorithm. The percentage rate of feasible timetables generated (see Figure 5.16) is stable across the values of σ. This is to be expected given the trails that σ intensifies. All trails are linear combinations of first and second-order reward. Intensifying the trail within pairwise sets of exams not only indicates second-order suitability but first-order as well. Consequently, this explains the robustness of the rate of feasible timetables. The summary statistics for this investigation are presented in Table 5.14.

σ	HEC			EAR			TRENT		
	Av	*Best*	*Best 10*	*Av*	*Best*	*Best 10*	*Av*	*Best*	*Best 10*
1	1361.15	837	918.36	1703.72	1265	1421.66	1452.27	1123	1195.92
5	1299.94	749	867.34	1553.03	1176	1319.64	1382.62	1074	1135.46
10	1245.77	756	844.66	1551.18	1081	1282.02	1303.07	920	1069.82
15	1264.16	745	863.26	1580.27	1208	1353.72	1387.21	1094	1156.76
20	1262.54	724	851.16	1411.05	1105	1180.82	1335.73	1013	1113.00
25	1064.28	736	819.76	1457.45	1080	1248.84	1341.35	1050	1106.14
30	1165.98	775	833.60	1457.91	991	1211.58	1304.12	986	1079.00
35	1151.17	704	811.46	1421.51	1049	1166.78	1370.32	1093	1147.10
40	1132.00	803	830.70	1461.58	1142	1258.14	1326.47	1001	1092.88
45	1105.61	782	806.70	1426.75	1110	1220.10	1301.70	1019	1082.66
50	1077.41	713	776.34	1478.16	1159	1264.80	1342.36	989	1097.82

Table 5.14– Second-Order Statistics describing influence of σ

Figure 5.16 details the percentage of feasible timetables across the three data sets for a range of σ.

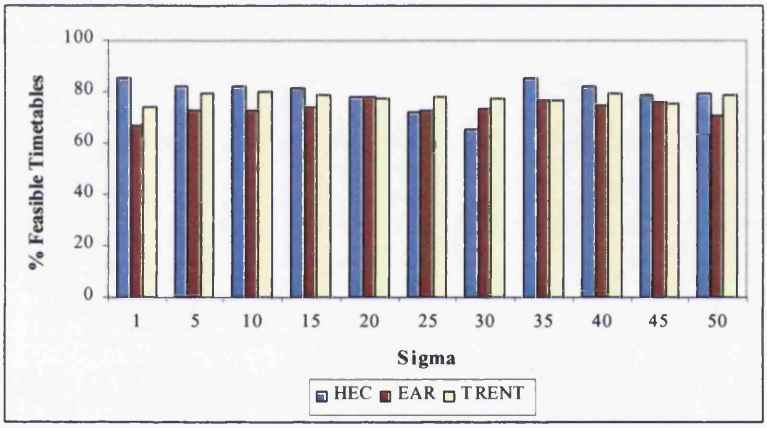


Figure 5.16 – Percentage of Feasible Timetables across Trail Intensification Weight σ

To avoid wasted precursory investigation, generalised settings will be used. Hence, the finalised parameter settings are as follows:

Parameter	Setting
e	15
$100+\delta_2$	750
σ	30

Table 5.15– Robust Elitist Settings

These settings have been used and run on the data sets. The results are presented in Table 5.16.

	Av	Best	Best 10	Feasible
HEC	935.66	673	711.28	76.84
EAR	1296.94	1037	1104.02	62.83
TRENT	1259.89	985	1039.48	71.45

Table 5.16– Second-Order Statistics for Robust Elitist Settings

The results are more than satisfactory and indicate that the elitist settings are robust. They will subsequently be known as *Robust Elitist Settings*. The statistics in Table 5.16 are considerably superior to those generated through non-elitist systems and are only bettered by some results linked with the usage of higher values of the second-order bias parameter δ_2. However, as we have seen above, the first-order quality of constructed timetables suffers when δ_2 increases and explains the setting chosen, $100+\delta_2=750$.

To adapt δ_2 to the disparities of scale between data sets and to construction difficulties, a management system that fine-tunes δ_2 is suggested. This parameter could vary according to the level of feasibility, the second-order performance of the search or the width of exploration. This concept is discussed at greater length in Chapter 7.

5.5.3.3 Additional Second-Order Bias

Section 5.5 has shown that the use of trail intensification has a positive effect upon solution quality. The approach discussed in Section 5.5.3.1 has the potential to produce good second-order solutions but neglects the first-order priorities of the problem. Meanwhile, Section 5.5.3.2 seems to address the balance between the first and second-order priorities. In this section, we attempt to improve the second-order fitness of the results achieved in Section 5.5.3.2 by incorporating a second list as in Section 5.5.3.1.

The work detailed in Section 5.5.3.1 is used as a basis and s ants are placed in a second 'elite' list. Membership is according to the second-order score of a timetable rather than its combined trail (first list). The aim is to recognise pairwise sets of exams that could contribute towards feasible low second-order conflict timetables. The set of 15 elite ants, e, from the previous section remain and we now vary s, the size of the second elite set. The parameters used for this investigation follow the *Robust Elitist Settings*.

Here the influence of varying s is tested and the results are presented in Table 5.17.

s	HEC			EAR			TRENT		
	Av	*Best*	*Best 10*	*Av*	*Best*	*Best 10*	*Av*	*Best*	*Best 10*
0	935.66	673	711.28	1296.94	1037	1104.02	1259.89	985	1039.48
5	899.60	671	697.64	1251.39	1002	1074.37	1302.62	1046	1076.24
10	**890.63**	632	695.38	1310.63	1104	1125.60	1261.05	992	1058.02
15	1039.67	696	766.36	**1156.04**	**932**	967.55	1283.76	988	1059.10
20	915.60	**620**	691.48	1368.80	1115	1159.45	**1247.25**	962	1036.86
30	1066.39	705	739.02	1388.59	1049	1160.32	1263.98	**946**	1006.32

Table 5.17– Second-Order Statistics describing the range of s

It can be seen that incorporating a second elite list does produce occasional better results, but the improvement is not as significant as anticipated. The best *Best* results have been bolded and it is observed that relatively large *s* is needed to make an impact. For HEC, the best *Av* solution is seen when *s=10* and the best *Best* at *s=20*. However, it is worth noting that the results achieved for *s=15* is no superior to *s=0*. For EAR, both the best *Av* solution and the best *Best* are observed when *s=15*. But, there is no real pattern across *s*. Typically, for EAR, the results returned at *s=10* and *s=15* are inferior to *s=0*. For TRENT, the results are rather indifferent, but marginally superior for larger *s*. The percentage of feasible timetables across *s* is presented in Figure 5.17.

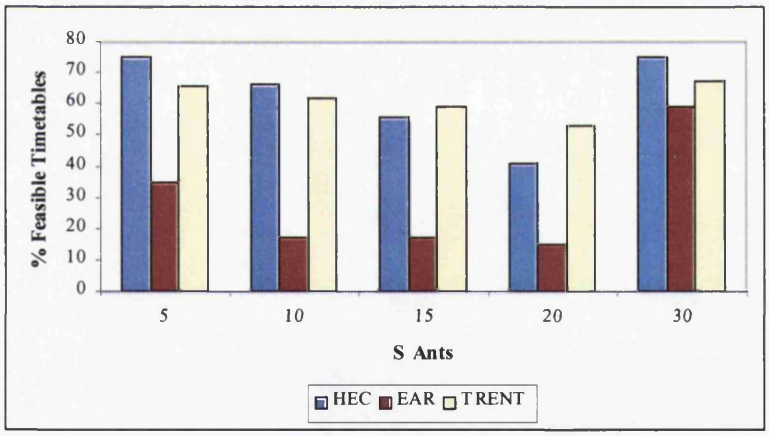


Figure 5.17 – Percentage of Feasible Timetables across S Ants

For all data sets, the distribution of the percentage of feasible timetables across *s* exhibits a bi-modal like distribution. Higher percentages are observed at *s=5* and *s=30* and lower rates between these values of *s*. An intuitive reason is as follows. As *s* increases, greater reward is placed on elite solutions chosen solely on second-order scores and thus, detracting from the first-order problem and consequently, feasibility rates suffer. In section 5.5.4.1, increasing the number of elitist ants had a detrimental effect on feasibility rates. However, this was not observed for bigger *s* within this section. Increasing the size of the second-order elite list limits differentiation between solutions and since more focus is also

placed on feasibility (through the ERF) the ants do not get trapped within infeasible regions of the solution space.

Section 5.5 has shown that trail intensification improves solution quality significantly, relative to what results have been observed previously. Stützle and Hoos (1996) stressed that elitism is necessary to enhance solution quality when using the ant system. It was demonstrated that the introduction of an *Elitist Reward Function* with an additional second-order bias δ_2 not only improves the second-order quality of the timetables generated but also maintains feasibility. In addition, a set of parameters were obtained that were robust to all datasets evaluated. The elite list was constructed according to the e best global solutions, however some authors, e.g. White et al. (2003) and Stützle and Hoos (1996), do claim that better search environments can be obtained through classifying elite on a local basis. However, it was observed through preliminary tests that quicker convergence was possible with global elitism and importantly, better solution quality within the allocated 100 cycles. If extended runs were permitted then the use of local elitism could prove beneficial given the additional diversity that it would yield. Alternatively, a system that alternates between global and local elitist lists has the potential to succeed. However, subsequent work within this chapter and Chapter 6 aims to increase the diversity of ant searches and therefore, fully evaluating the efficiency of local-influenced elitist lists is not felt warranted.

5.5.4 Ant Synergy

Figures 5.18-5.20 provide evidence of the positive communication between ants with respect to second-order when using the *ERF* with *Robust Parameter Settings*. The average and running best scores against time (cycle) for feasible constructions are plotted for each of the three data sets. These statistics have been averaged across five runs. Each line graph begins at the cycle when all five runs return at least one feasible timetable.

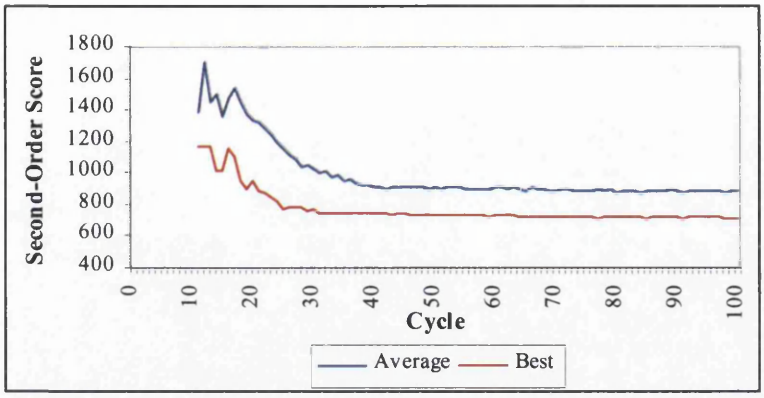


Figure 5.18 – Average and Best Second-Order Score versus Cycle plot for HEC under Elitist conditions

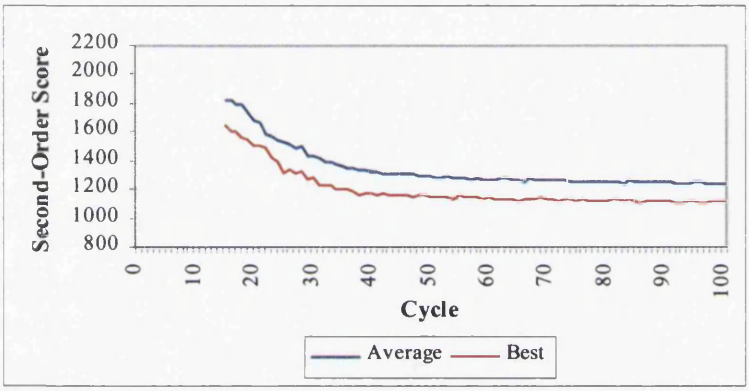


Figure 5.19 – Average and Best Second-Order Score versus Cycle plot for EAR under Elitist conditions

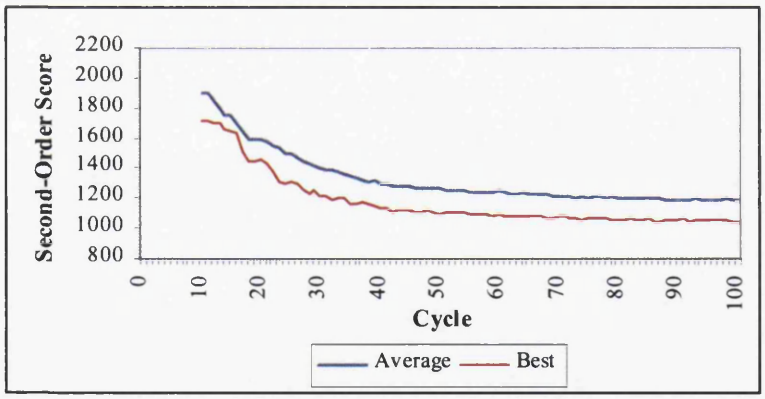


Figure 5.20 – Average and Best Second-Order Score versus Cycle plot for TRENT under Elitist condition

Figures 5.18-5.20 show the strong learning capabilities of the ants that use the *ERF* method. Solution improvement continues for longer and is more intense than previously experienced (plots not presented for other systems e.g CRF).

5.6 Ant Convergence

A characteristic of the ant system is that agents do not generally converge to a common timetable. The inherent probabilistic nature of the algorithm reduces the possibility of stagnation occurring. However, as a search matures, the diversification of the ant investigation is reduced. Concentration is placed on a subset of the solution space and consequently, a greater number of common solution characteristics between timetables occurs. The underlying aim of such search processes is to find that balance between exploration and exploitation. This balance has been examined through the use of two studies. Firstly, we will take a look at the standard deviation of an example run from each of the three data sets. Secondly, we will investigate the branching possibilities as the search matures.

5.6.1 Standard Deviation

Figures 5.21-5.23 plot the cyclic standard deviations of the second-order scores for two search environments. The first search process uses the Elitist Reward Function (ERF) under the *Robust Elitist Settings* (i.e. the number of elitist ants, e, is 15, while the second-order bias $100+\delta_2$ is 750 and the trail intensification weight σ is 30). The second search environment uses the Combined Reward Function (CRF) with a second-order component weight, δ, of 100. These figures show that the levels of diversification of the different search types and demonstrate whether the ants are converging on the same solutions.

Generally, the standard deviation rate reduces until some stage and then dampens. These characteristics indicate that the search becomes more focused as it matures and this can be attributed to the exploitation of 'good' feedback by the ants. For HEC and TRENT, the cyclic standard deviations are lower for the ERF than the CRF search environment. This observation suggests that the use of elitism reduces the options of the ants since extra reward is placed on an 'elite' subset of solutions and subsequent ant decisions are inclined towards these more heavily rewarded solution characteristics. Reducing the number of choices for the ants is not necessarily detrimental and can lead to better solution quality as greater exploitation is performed in certain subsections of the solution space. Encouragingly, Figures 5.21-5.23 show that the ants do not converge to the same solution types and there is at least some search variability at mature stages of the search process.

The average number of students taking each exam (*#Studs/Exam*) is 120.03, 42.77 and 57.08 for HEC, EAR and TRENT respectively. This ordering of *#Studs/Exam* corresponds to the scales on the *Std Dev* axis on each of the Figures below. The disparity between the plots within each Figure could be attributed to the average change in second-order costs per exam move. For example, with HEC, large deviations associated with the CRF system indicate that a large range of second-order scores is obtainable, given *#Studs/Exam*, if no exploitation characteristics (with respect to second-order) are exhibited. Therefore, narrowing the solution space that is explored through the ERF leads to smaller changes between solutions and consequently, a drastic drop in standard deviation is experienced. Similar statements, but to a lesser extent, can be made regarding TRENT. However, with EAR, the standard deviations of the ERF and CRF are similar. This may be attributed to the difficulty of the problem and consequently, the feasible timetables that are produced may be closer in characteristics when compared to the other data sets. This theory is supported by the *Entropy* values presented in Section 4.4.1.3, which show that EAR timetables have the smallest diversity.

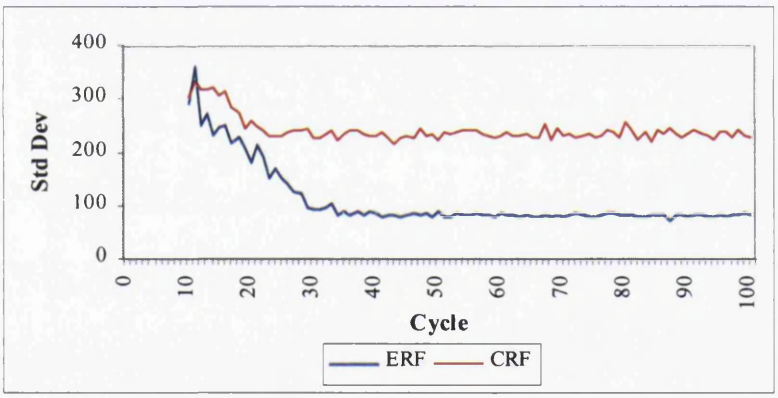


Figure 5.21 – Cyclic Standard Deviations of Second-Order Scores for ERF and CRF for HEC.

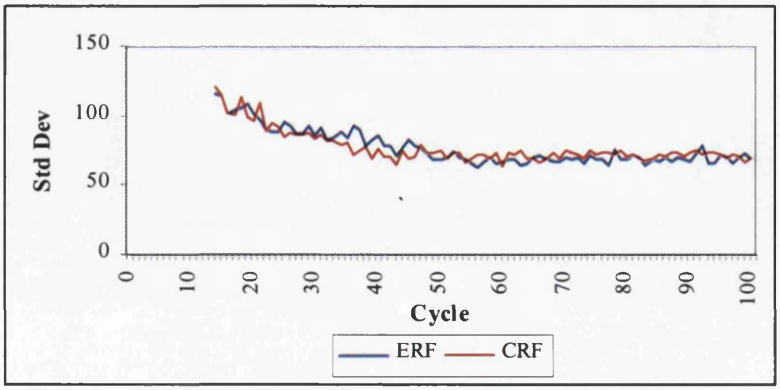


Figure 5.22– Cyclic Standard Deviations of Second-Order Scores for ERF and CRF for EAR.

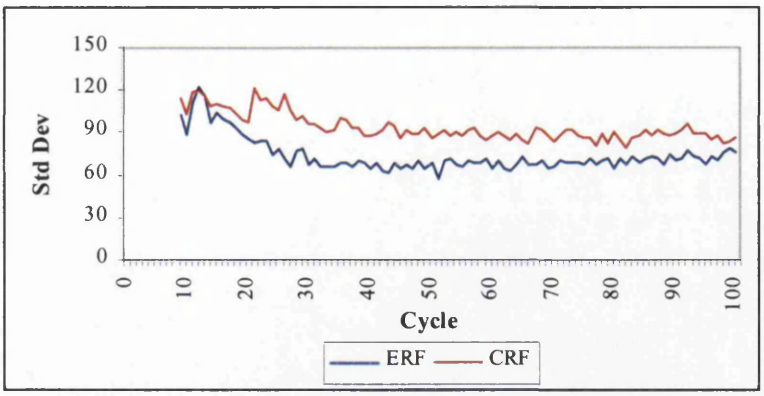


Figure 5.23 – Cyclic Standard Deviations of Second-Order Scores for ERF and CRF for TRENT.

5.6.2 Branching Factors

To demonstrate the focus of the search, we will examine the mean branching factor $\overline{\lambda}$, Gambardella and Dorigo (1995). The aim of the search process is to obtain a balance between exploitation and exploration. Stützle and Hoos (1997) suggested that best solutions are found near stagnation environments since the ants concentrate on a small subset of the solution space. If $\overline{\lambda}$ represents the dimension of the solution space that is being investigated, it is claimed by the above authors that lowering $\overline{\lambda}$ to some mark is desirable. The mean branching factor was defined formally in Chapter 2 (Pages 40 – 41) and will be revisited here.

Let $AQ_{max}(r,s)$ and $AQ_{min}(r,s)$ be the largest and smallest trail values on all feasible pairwise vertices (r,s), where $\delta_r = AQ_{max}(r,s) - AQ_{min}(r,s)$. The mean branching factor for a vertex r is represented by the number of vertices s, with a pairwise trail score greater than $\lambda.\delta_r + AQ_{min}(r,s)$. It was suggested by Stützle and Hoos (1997) that setting $\lambda = 0.05$ is appropriate and will be used here. Figures 5.24-5.26 plot the cyclic mean branching factors for the ERF and CRF.

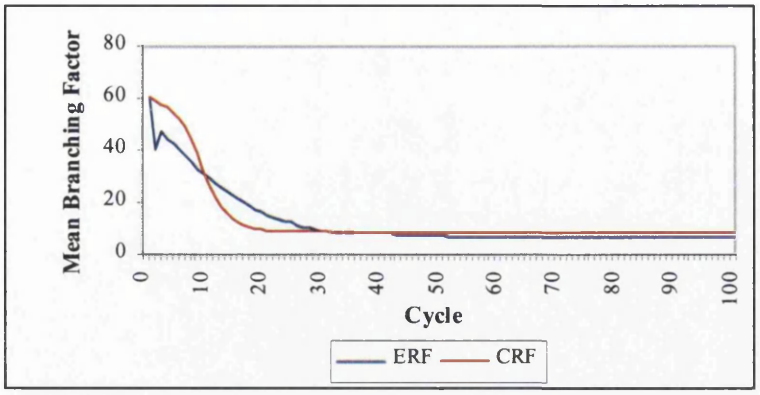


Figure 5.24 – Cyclic Mean Branching Factor when using ERF and CRF for HEC.

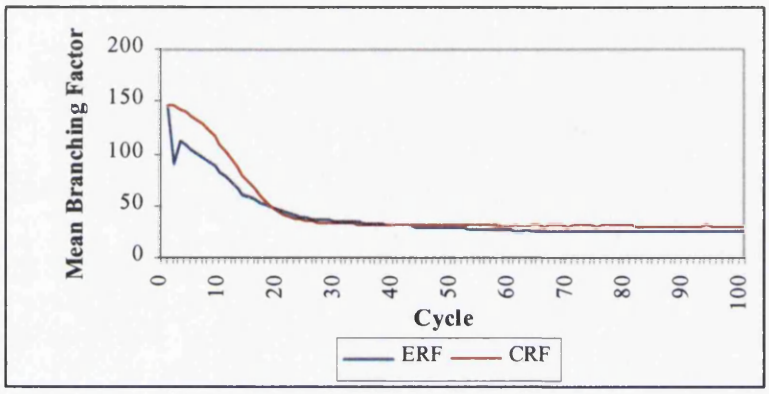


Figure 5.25 – Cyclic Mean Branching Factor when using ERF and CRF for EAR.

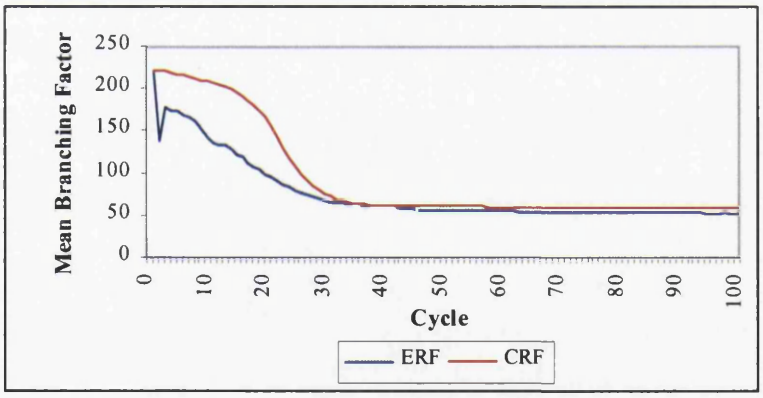


Figure 5.26 – Cyclic Mean Branching Factor when using ERF and CRF for TRENT.

Figures 5.24-5.26 show that the dimension of the solution space reduces as the search matures for both the *ERF* and the *CRF* related environments. For all data sets, the mean branching factors for the *ERF* and *CRF* are similar from roughly the 30^{th} *cycle*. Prior to this mark, the mean branching factors for EAR and TRENT are consistently lower for *ERF* than the *CRF*, which indicates that the area of the solution space that is explored by the ants is smaller during the early phases of the search and (potentially) more focused. This

can lead the ants to investigate better quality regions of the solution space as the search matures due to the more efficient investigations during the earlier stages of the search. A slightly different pattern occurs with HEC, the ERF cyclic mean branching factor is lower from the onset of the search. However, by approximately *cycle 10*, the *CRF* system has the lower mean branching factor and this continues until *cycle 30*. However, the lower branching factor suggests that the search is more focused during the infancy of the search and this is significant with respect to obtaining overall better solution quality.

Each experimental run that has been performed (on the *ERF* and the *CRF*) is analysed here (Table 5.18) to show the benefit of the *ERF* over the *CRF* and consequently, emphasises the overall benefit of a proficient search process during the early phases of the search. Along with the fundamental statistics in Table 5.18, the *Best* solution at the 30^{th} *cycle* is also recorded. This statistic has been included to assess the improvement witnessed after the *dampening point* (as seen in Figures 5.24-5.26).

		ERF				CRF			
	Run Number	*Cycle of 1st feasible*	*Overall Best*	*Best found @ cycle*	*Best @ 30th cycle*	*Cycle of 1st feasible*	*Overall Best*	*Best found @ cycle*	*Best @ 30th cycle*
HEC	1	8	693	49	698	10	945	34	947
	2	9	770	47	792	8	1013	20	1013
	3	10	712	61	776	9	987	20	987
	4	6	673	65	765	10	912	42	946
	5	6	694	85	752	4	969	36	975
EAR	1	14	1095	93	1198	-	-	-	-
	2	12	1144	65	1314	13	1404	85	1490
	3	11	1150	97	1349	13	1319	75	1354
	4	13	1037	54	1182	14	1568	28	1568
	5	14	1038	98	1206	12	1381	56	1463
TRENT	1	7	1098	99	1265	3	1438	34	1465
	2	7	1069	87	1248	2	1445	15	1445
	3	9	994	92	1139	2	1455	11	1455
	4	8	983	87	1188	3	1289	79	1399
	5	7	991	87	1227	3	1413	18	1413

Table 5.18 – First and Second-Order Statistics describing the effectiveness of the ERF over the CRF

Table 5.18 demonstrates the benefit of using the *ERF* based system ahead of the *CRF*. The quality of the best solution found is significantly better for *ERF* than *CRF*. It can be seen that, in all cases, the best solution at the 30^{th} *cycle* for the *ERF* is better than the overall best solution for the *CRF*. It is shown that improvements of the best solution after the 30^{th} *cycle* are commonplace for *ERF* but only occur in 8 out of the 15 instances for the *CRF*, with only 4 of these occurring after the 50^{th} *cycle* (suggesting that the second half of these searches are often obsolete).

The use of the first-order component in the *ERF* allows for the maintenance of feasibility and, in some cases, improves feasibility rates. The 1st run of EAR is an example of this. The *CRF* system does not generate one feasible timetable, while *ERF* leads to the construction of (at least) some feasible timetables. The ability of the *ERF* to encourage more feasible timetables will be discussed further in Section 6.4. Additionally, the timing of the observation of the first feasible timetable is noted. It can be seen that the statistics for *ERF* and *CRF* are similar for HEC and EAR. However, there is a disparity with TRENT, but the trade-off for slower feasible return is the significant improvement of second-order quality.

This section has shown that the ants that construct timetables during the latter cycles will benefit considerably if more focused searches are performed within the infancy of the search. Placing more reward on 'elite' solutions narrows the investigations of the ants, thus leading to better solutions in the short term. Subsequent ants then exploit (as the mean branching factor gets smaller) the trails in desirable areas of the solution space and these actions allow for the further improvement of *best-found* solutions during later cycles.

5.7 Influence of Timeslot Vertices

Section 5.2 suggested that incorporating a trail between exams and timeslots offers additional search potency and such trail has been incorporated in this Chapter. In this section, evidence confirming this statement is provided. The *ERF* with *Robust Elitist Settings* system is used for three search environments. The first updates the trails between pairwise exam sets and the pairwise exam-timeslot sets (labelled E1), which conform to the system used in Section 5.5.3.2. The second environment only updates the trails between pairwise exam sets (labelled E2) and the third only updates the trails between pairwise exam-timeslot sets (labelled E3). Results for these environments are presented in Table 5.19.

	Environment	Best	Feasible
HEC	*E1*	673	76.84
	E2	770	68.53
	E3	804	28.00
EAR	*E1*	1037	62.83
	E2	1216	64.52
	E3	-	0.00
TRENT	*E1*	985	71.45
	E2	1210	67.50
	E3	1150	64.56

Table 5.19 – Quality Statistics for Three Trail Update Environments

Table 5.19 presents the following observations. On a first-order level, E1 is marginally superior to E2 and both E1 and E2 significantly outperform E3. This indicates that the relative position of exams is more important that the absolute position when regarding the feasible timetables. Meanwhile, it can be seen that the second-order results regarding E3 are comparable with E1 and E2. The exception here is the EAR dataset due to the construction of no feasible timetables. Importantly, the results presented in Table 5.19 emphasise the benefit of accumulating feedback regarding both the relative and absolute positions of exams in timetables. Therefore, environment E1 will be used in all future studies.

5.8 Weighted Construction Heuristic

As detailed previously in this thesis, the choice of the exam to allocate depends primarily on the balance between trail accumulation and the level of visibility, along with the inherent randomisation. Exam selection is also influenced by a constant, which represents the impact on the second-order score of the timetable by allocation of that exam to some timeslot.

In Chapters 3 and 4, it was proven that the choice of construction heuristic has a big influence on the first-order capabilities of the algorithm. It was deemed that the most appropriate individual heuristic (when used along with ants) was type *C* (see Chapters 2 and 3 for interpretation and definitions of construction heuristics), which evaluated the degree of an uncoloured vertex within the uncoloured subgraph. It is a dynamic structure and the degree of each uncoloured vertex is updated after the colouring of a vertex. However, in this section we modify heuristic *C* with respect to the examination timetabling problem and weight the degree of an exam by the number of student clashes with other

exams belonging to the uncoloured subgraph. Arani and Lofti (1989) ordered exams according to their weighted degree and performed exam-timeslot allocations in the manner of a traditional heuristic, e.g. Leighton (1979), to obtain an initial solution in their three-phase approach to the examination timetabling problem. The authors described the weighted degree as in Equation 5.12.

$$w_i = d_i \sum_k a_{ik} \quad (5.12)$$

where d_i is the degree of exam i and $\sum_k a_{ik}$ is the total number of students that take exam i and all other exams in conflict with exam i. With respect to this investigation, the aim of this heuristic is to deal with higher weighted degree exams earlier in the timetable. Therefore, it is idealised that higher clash exams will be allocated to the *Odd* timeslots (with respect to the *OddEven* construction process) and thus will increase the possibility of a reduction in second-order conflicts. All other components of the algorithm remain unchanged. This construction heuristic will be used to quantify the desirability score of a vertex at some vertex-colour decision point. This component will again be raised to the bias parameter β and β is now varied to assess influence on solution quality. The trail bias α will again be set at 2. Table 5.20 displays the results.

		Robust Elitist Settings	*Weighted Construction Heuristic*			
	β	1	1	2	3	4
HEC	*Av*	**935.66**	1127.48	1185.37	1210.28	1205.24
	Best	**673**	**607**	649	624	638
	Best 10	**711.28**	672.02	700.44	722.50	726.14
	Worst 10	**2237.92**	2278.84	2316.02	2335.92	2354.40
EAR	*Av*	**1296.94**	1328.09	1362.81	1358.13	1509.69
	Best	**1037**	**977**	1011	1014	1125
	Best 10	**1104.02**	1044.04	1066.54	1086.20	1187.74
	Worst 10	**1788.42**	1910.66	1905.62	1918.38	1912.46
TRENT	*Av*	**1259.89**	1258.30	1258.04	1339.41	1380.71
	Best	**985**	989	**900**	910	985
	Best 10	**1039.48**	1026.14	957.84	947.50	1071.84
	Worst 10	**1801.48**	1802.92	1809.84	1840.74	1801.28

Table 5.20 – Second-order statistics for heuristic C and Weighted Construction Heuristic under Preferred Elitist Conditions

For the purpose of comparison, the results under the *Robust Elitist Settings* using heuristic type *C* are also presented in Table 5.20. We note that better *Best* results can be obtained through the weighted construction heuristic method despite poorer average second-order

scores. Since the *Best 10* and *Worst 10* scores are competitive via the weighted construction heuristic, it suggests that the ants explore a greater area of the solution space, which is desirable. Generally, the influence of β is reasonably small. For HEC and EAR, $\beta=1$ is the best result, while $\beta=2$ is the most appropriate setting for TRENT ($\beta=1$ has comparable solution quality with those achieved via construction heuristic *C*). Only for EAR, when $\beta=4$, do we experience a significant deterioration in solution quality. Figure 5.27 presents the percentages of feasible timetables for range of β for the three main data sets is presented. The first-order capabilities of heuristic *C* is also charted for comparison purposes.

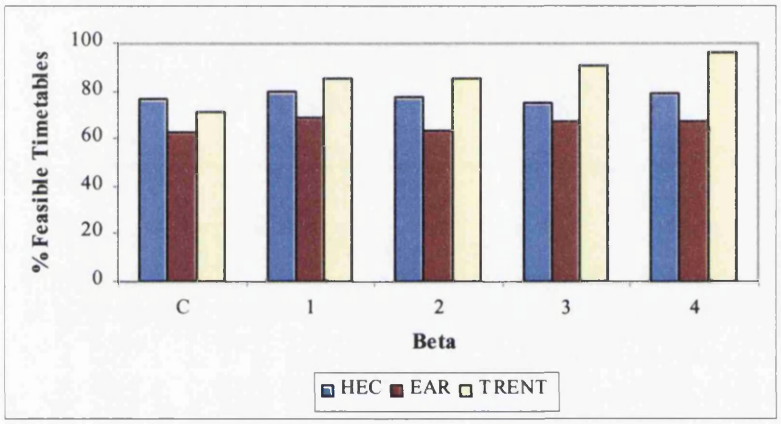


Figure 5.27 - Percentage of Feasible Timetables across Bias Parameter β - Weighted Construction Heuristic

The percentage of feasible timetables is fairly consistent across β. It has been demonstrated in Chapter 3 that increasing any power exponent leads to slower runtimes. Therefore, maintaining lower β is preferred given that there is no real trade-off with solution quality. Given this, $\beta=1$ will be used for all data sets.

5.9 Limiting Exploitation

Even though the use of elitism does not generally result in stagnation, the method does reduce the area of the solution space that is explored. This is desirable to some level, but it is advisable to encourage as much diversification as possible while retaining the focus on better solutions. An approach to tackle this was suggested by Stützle and Hoos (1997) who detailed the MMAS that imposed lower, t_{min}, and upper, t_{max}, limits for trail levels. The authors proved that such a controlled search environment could produce some very good solutions. The trail limits were modified dynamically when certain search characteristics,

such as stagnation, were recognised. The authors also suggested that resetting all trails to t_{max} when appropriate led to superior search conditions for the ants. The use of trail limits is imperative when using MMAS due to the use of only the best ant to update trails, which encourages reduced diversity and relatively poor final solution quality. A review of MMAS is presented in Chapter 2. The use of a trail limiting restrictive mechanism will be used within this section to encourage greater diversity. Focus is placed on imposing an upper bound since preliminary tests demonstrate that negligible benefit can derive from the use of a lower bound.

At the end of each cycle, the average highest trail measure was recorded and has been plotted below (Figure 5.28). This chart provides an indication of the range of trail levels that are present and the highest level of trail that may be required to achieve the results that have been presented in Section 5.7. Figures 5.18-5.20 illustrate the improvement of solution quality over time, for the non-weighted construction heuristic, and the averaged best plots show that the probability of observing a new best solution is reduced after the early phases of the search. On Figure 5.28, this is represented by the low gradient part of the curves. In this section, we will experiment with t_{max} levels below the point where trail levels dampen. As an aside, let us note the limited disparity between the trail levels of the data sets. The HEC trail curve is predominately larger due to the higher reward that is passed to the trail (in comparison to EAR and TRENT) due to the good volume of timetables and relatively lower second-order scores. Meanwhile, the trail level for TRENT is greater than for EAR due to the lower unallocated exam count and consequent higher reward. This is the main difference between the data sets as the second-order scores are not too dissimilar.

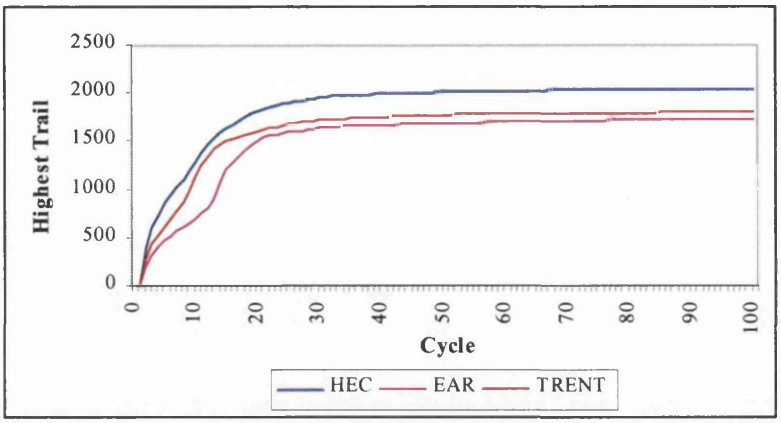


Figure 5.28 – Cyclic Highest Trail Levels

It should also be noted that varying the number of ants per cycle does not have a significant influence on the highest trail levels.

Various upper trail levels are evaluated here and dealt with on a static basis. Upper limits of 50 units apart were considered. Results for selected t_{max} are as follows.

	HEC			EAR			TRENT		
T_{max}	*Av*	*Best*	*Best 10*	*Av*	*Best*	*Best 10*	*Av*	*Best*	*Best 10*
-	*1127.48*	*607*	*674.02*	*1328.09*	*977*	*1044.04*	*1258.30*	*989*	*1026.14*
200	1394.79	780	890.96	1648.08	1192	1334.98	1612.88	1230	1371.14
300	1315.70	733	814.64	1525.09	1045	1197.56	1486.29	1078	1172.52
400	1071.26	604	682.82	1358.45	977	1083.26	1341.90	943	996.24
500	961.44	640	672.46	1281.90	939	**1024.50**	1268.33	921	977.72
550	**868.24**	**584**	**614.02**	1314.31	982	1066.04	1233.91	936	972.24
600	1028.14	638	672.46	1282.46	**917**	1028.56	1276.72	952	1006.76
700	941.63	608	650.46	**1276.23**	1006	1057.32	1209.87	908	960.20
800	882.89	657	699.22	1304.21	959	1086.24	**1190.45**	914	**946.78**
850	947.13	694	719.42	1296.04	953	1060.60	1212.10	**878**	978.30
900	887.92	673	703.50	1341.44	1050	1123.22	1196.62	889	968.96
1000	911.61	704	742.62	1317.11	972	1097.38	1211.12	923	996.22

Table 5.21– Second-Order statistics across range of t_{max}

The italics row refers to runs where no trail limit was imposed and the weighted construction heuristic was biased to $\beta=1$. Imposing an upper limit on the trail does improve solution quality. However, this depends on the appropriate selection of t_{max}. Low t_{max} does not allow significant differentiation between pairwise exam sets and consequently, the search process experiences a lack of focus and solution quality suffers. Table 5.21 supports this claim. Across the data sets, $t_{max}=200$ and *300* leads to poorer solution quality than the unrestricted trail model. However, higher measures of t_{max} do lead to better solutions, which support restricting trail levels. The bolded statistics refer to the

best results found. For HEC, t_{max}=*550* leads to the best 'one-off' and average solution, while, for EAR, t_{max}=*600* generates the best individual solution and t_{max}=*700* produces the best average solution, however, average solutions for t_{max}=*500* and *600* are in the same region. For TRENT, higher t_{max} is needed to find the best search conditions. Settings of *800* and *850* generated the best average and minimum solutions respectively. Generally, as t_{max} approaches *1000*, solution quality deteriorates again.

Imposing t_{max}=*550*, *600* and *850* seems appropriate for HEC, EAR and TRENT respectively. We note here that the preferred setting of t_{max} seems to increase with the size of the problem and consequently, it is suggested to exploit this relationship when attempting to deduce t_{max}. A little trial and error could be then performed to deduce more accurate t_{max} settings.

Stipulating an upper limit t_{max} does have a major influence on the first-order capabilities of the algorithm. Figure 5.29 presents a sample of t_{max}.

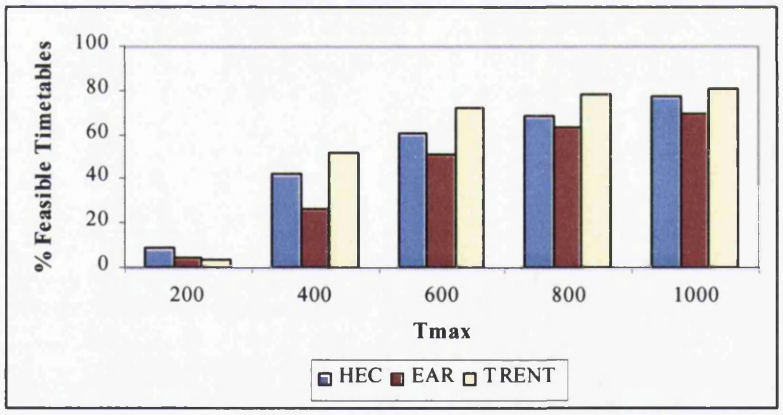


Figure 5.29 – Percentage of Feasible Timetables for range of t_{max}

The relationship between t_{max} and the percentage of feasible timetables is clear – the higher the value of t_{max}, the greater the volume of feasible timetables. In Section 5.6.2, we examined the relationship between the area of the solution space that is examined and the quality of solutions found. Here, we measure the influence of imposing trail limits on the dimension of the solution space. The cyclic mean branching factor is again used to measure the width of exploration and compares the *ERF* environment with the *ERF* environment that imposes trail limits (t_{max}=x, where x is the upper trail limit). The results are illustrated in Figures 5.30-5.32.

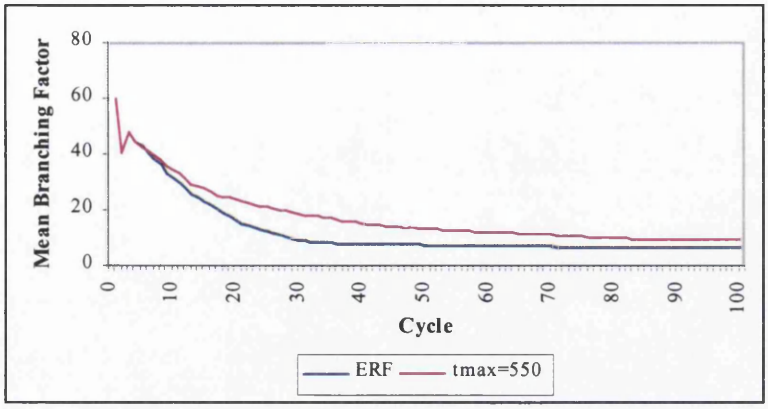


Figure 5.30 – Cyclic Mean Branching Factor for ERF and ERF when t_{max}=550 for HEC

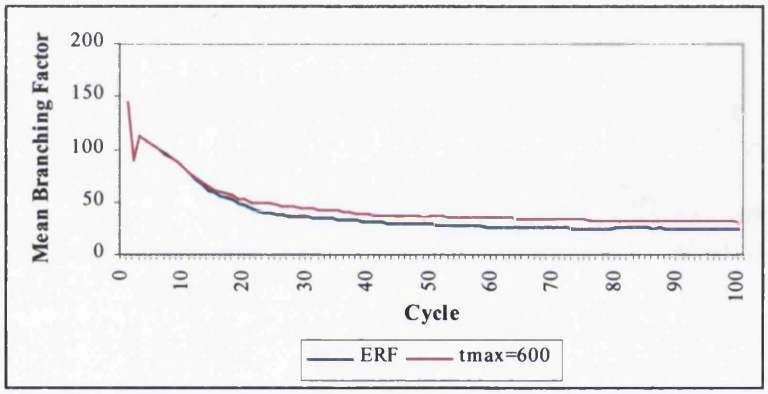


Figure 5.31 – Cyclic Mean Branching Factor for ERF and ERF when t_{max}=600 for EAR

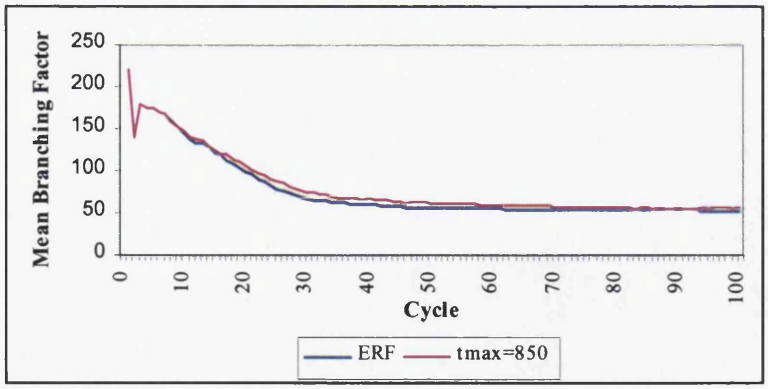


Figure 5.32– Cyclic Mean Branching Factor for ERF and ERF when t_{max}=850 for TRENT

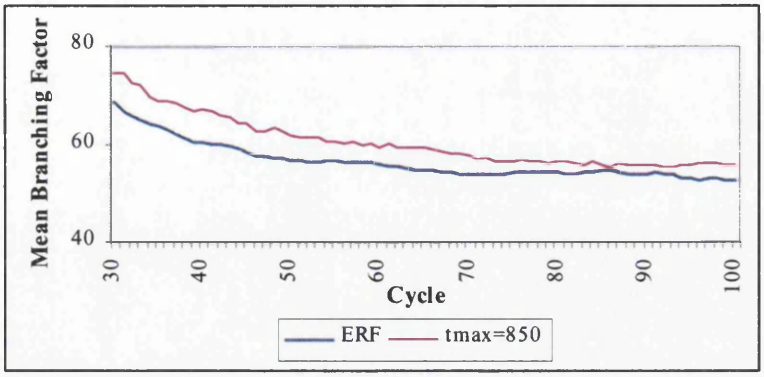


Figure 5.33 – Cyclic Mean Branching Factor from 30th cycle for ERF and ERF when t_{max}=850 for TRENT

Figures 5.30-5.32 present mean branching factors across the independent runs for all 100 cycles. These Figures demonstrate the inherent extra exploratory power that exists by imposing trail limits. This is shown by higher mean branching factors towards the end of the search. The *ERF* and $t_{max}=x$ systems exhibit similar behaviour during the infancy of the search. This is advantageous since narrower explorations in the early stages of a search lead to superior results (evidence has been presented in Section 5.6.2, which compared *CRF* and *ERF* systems). However, the *ERF* and $t_{max}=x$ systems differ from approximately the *10th cycle* for the data sets, with the latter demonstrating wider exploration. This is the reason behind the superior results. The disparity of the mean branching factors between the *ERF* and $t_{max}=x$ systems for EAR and TRENT do not appear significant but this can be attributed to the scales used on these Figures. To illustrate the true difference between the systems, an illustration of the mean branching factors during the latter stages of the searches for the TRENT data set has been presented in Figure 5.33. It is demonstrated that an exploration disparity between the systems does exist and greater exploration is performed, thus leading to superior results.

5.10 Degree of Saturation

Until this juncture, the heuristic type *C* has been used extensively due to the superior number of solutions that it generates. However, the basic structure of the Degree of Saturation (DSatur) heuristic is a more suitable framework when considering second-order conflict. The selection of exam then timeslot allows greater flexibility and consequently,

gives one the ability to (potentially) manipulate exam-timeslot relationships. Costa and Hertz (1997) detailed two variants of DSatur, which have been described in Chapter 2.

Work performed in Chapters 3 and 4 showed that construction heuristic G performed consistently better than *H* and, the use of *H* on highly constrained examination timetabling problems would result in volumes of infeasible timetables. This observation is unfortunate as the inherent structure of *H* provides one with the greater range of choice of colour. To remind the reader of the difference between the construction heuristics, a brief description of *G* and *H* is provided here. Heuristic *G* allocates exams to the minimum feasible timeslot, c_{min}, biased according to the cumulative trail between an exam and the exams belonging to c_{min}, while *H* selects an exam and then allocates that exam to the timeslot (from a set of feasible timeslots) biased according to the accumulated trail.

Despite the inferiority of *H* (in comparison to *G*) with respect to solution quality, it has been used as the framework for the first DSatur related examination timetabling construction procedure.

5.10.1 Greedy DSatur

In this section, the population of ants is removed from consideration and the structure of *H* is exploited. The approach is defined as follows. The probability of selecting exam *e* (p_e) is weighted by the degree of saturation, $[v(b[k-1], e)]$, and biased by β (in order to increase the marginal influence of the degree of saturation). It is defined as

$$p_e = \frac{[v(b[k-1], e)]^{\beta}}{\sum_{r \in T} \{[v(b[k-1], r)]^{\beta}\}} \quad (5.13)$$

with *T* referring to the set of exams available for allocation at some exam-timeslot decision point.

After the selection of exam *e*, it is assigned to timeslot *t* that contributes the minimum amount of second-order conflicts to the timetable.

This approach has some pertinent drawbacks. Due to the absence of a controlling driving force i.e. trail; the prospect of obtaining a feasible timetable is drastically reduced. Since such a result is the overriding concern it will limit the attractiveness of this strategy. Experiments showed that not limiting the number of exams available for allocation produces very poor results. These results are not re-produced here. Achieving feasible timetables was rare and this pointed towards the incorporation of some constraining condition that truncates the volume of exams free for scheduling. Here we employ a candidate list that allows a number *n* exams to be considered at an exam allocation decision point. The unscheduled exams are sorted according to degree of saturation and only the exams with the higher saturation scores are permitted for allocation. The size of the candidate list has been varied and the results are displayed in Table 5.22

	HEC		EAR		TRENT	
n	*Av*	*Best*	*Av*	*Best*	*Av*	*Best*
1	1301.43	916	1976.76	1961	1870.60	1657
2	1152.08	697	1790.10	1271	1723.03	1236
3	1141.72	681	1741.54	1229	1644.23	1211
5	1144.33	734	1682.41	1377	1563.77	1171
10	-	-	-	-	1485.27	1173

Table 5.22 – Average and Minimum second-order scores for various n

The percentage of *Feasible* timetables for the experimental candidate list sizes for each of the main data sets is presented in Figure 5.34.

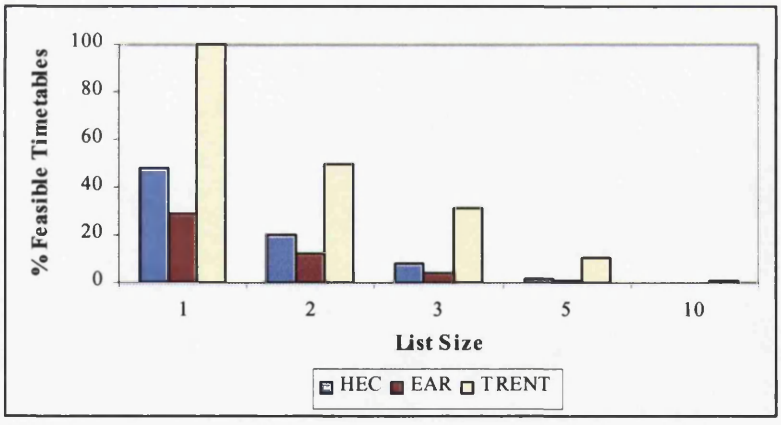


Figure 5.34 – Percentage of Feasible Timetables for various List Sizes (n).

Large lists are not worthwhile since the bias towards bigger saturation-degree exams is reduced and increases the potential of 'problematic' exams being dealt with towards the end of the construction process. It is shown that the size of list is required to be around 2 or

3 in order to produce a significant volume of feasible timetables, however, in consequence, the variation in timetables is limited. With respect to second-order conflict, lower list sizes generate better solutions. For HEC and EAR, a list size of 3 produces the best quality feasible timetable, while for TRENT, a list size of 5 does likewise. These 'best' results are superior to those presented in Table 5.1, but do not compete with elitist based methods.

To generate greater feasibility, the use of trail is required. It should be remembered though, that Chapter 3 illustrated that heuristic *H* was the weakest of all the construction procedures proposed and the feasible timetable count was low. Consequently, candidate lists were included to enhance feasibility rates as such modification was proven to be beneficial above. Again, the candidate list membership depends on the degree of saturation. After the exam is chosen, the choice of timeslot is biased by a combination of trail and an inverted partial second-order clash score, which corresponds to the level of additional second-order conflicts that the timetable will encounter by the allocation of that exam to that timeslot. The level of weight placed on the second-order clash score is robust and does not influence solution quality, although, incorporating a bias parameter γ (2 and 3 were experimented) does generate marginally better solutions. Overall though, the results are poor and consequently, a slightly different approach was attempted. RLF and DSatur type heuristics (*RLF_DSatur*) were mixed within each cycle. RLF heuristic type *C* has been shown to have the ability to produce feasible timetables, and through the appropriate second-order techniques, the ants clearly co-operate to enhance the second-order quality of the timetables. Within each cycle, a percentage of ants use the DSatur heuristic, while the remainder use RLF. Feedback contributions are made to the same trail and intuitively, the DSatur based ants would benefit from the trails laid by the RLF based ants. The percentage of DSatur ants within each cycle was varied and it was found that this proportion had a major role in determining the volume of feasible timetables. The higher the percentage, the lower the first-order capabilities, while lower candidate sizes encouraged greater feasible timetable scores, but lower variation within the timetable structures. If no candidate list structure is used then the feasible timetable count reduces to below 10%. This statistic has been generated for uncapacitated problems, thus tight seating capacities will lower that figure further.

The *PET* technique that was discussed in Chapter 4 was shown to increase the volume of feasible timetables. This approach did improve first-order statistics but, as a result, the lack of exploration did not lead to better quality feasible timetables.

5.10.2 DSatur Type 1 [C_{min}]

Chapters 3 and 4 showed that heuristic *G* was capable of producing very good first-order solutions. Heuristic *G* observes c_{min} and allocates exams to the lowest feasible colour. Here, we regard a manipulation of the structure of c_{min}. We observe Sec_{min}, which is the feasible timeslot with the lowest second-order conflicts.

The results do not compete with the previous best-found solutions and are not given here.

In conclusion, despite the potential of exploiting the structure of the degree of saturation heuristics, the weak first-order statistics associated with this approach are poor and indicate unsuitability as an examination scheduling solution method.

5.11 Two Populations

Until this juncture, only one population has been used to tackle the examination timetabling problem at any given time. In this section, we use more than one colony of ants to investigate the problem. Michel and Middendorf (1999), Doerner et al. (2001) and Gambardella et al. (1999a) all discussed models that used more than one population. These approaches are discussed in depth in Chapter 2. The use of a second ant colony introduces a second trail, which allows additional focus to be placed on particular objectives, typically, second-order conflict. At regular intervals (end of each cycle typically), information from the second trail will be injected into the first trail. The structure of this section is as follows. Firstly, two trails accumulate information with respect to the same reward function (*ERF*). This is performed for comparative purposes (for other dual population methods presented) and attempts to exploit the inherent randomisation within the algorithm. Secondly, the two populations search the solution space while observing different objectives. In this section we resort to a system that uses construction heuristic *C* and no trail limits.

5.11.1 Same Reward function

This investigation begins by employing two colonies (in parallel) that use the same feedback mechanisms. The aim is to exploit any favourable solution characteristics that are generated via a second population. The inherent randomisation means that the second population will certainly reach different solutions than the first colony. The stochastic element of ant algorithms can play a big role in solution quality and this influence is exploited here.

To avoid both trails converging on the same set of solutions, information exchange between the trails is not two-way. At each update stage, a subset of best solutions is passed from the 2nd trail to the 1st trail. Additionally, 'elite' ants with regard to both colonies are stored and extra replenishment is passed to the trails belonging to the 'elite' solutions. Let us denote e_1 as the number of elitist ants associated with the 1st trail and e_2 as the number of elitist ants that passes information to the 1st trail from the 2nd trail. The e_2 elitist ants also deposit extra (elitist-based) pheromone on 2nd trail solutions. These parameters have been chosen as follows: firstly, when $e_1=8$ and $e_2=7$, the original setting of *15 elitist ants* has been approximately evenly distributed between the two trails. Secondly, when $e_1=15$, $e_2=5$, $e_1=15$, $e_2=10$ and $e_1=15$, $e_2=15$, the original 15 ants associated with the 1st trail have been maintained and an additional set of 'elite' ants associated with the 2nd trail has been introduced. The size of this extra set is varied and the results are presented in Table 6.1. Setting $e_1=15$ and $e_2=n/a$ refers to the one-trail *Robust Elitist Settings* system.

	e_1	e_2	*Average*	*Best*	*Best 10*	*Feasible*
HEC	8	7	900.31	702	741.46	81.01
	15	5	919.07	**633**	734.06	74.39
	15	10	950.92	**656**	697.30	72.83
	15	15	913.29	**661**	718.56	62.51
	15	N/a	935.66	673	711.28	76.84
EAR	8	7	1299.97	1044	1110.88	62.52
	15	5	1307.76	1057	1122.62	64.69
	15	10	1294.34	1080	1122.94	64.78
	15	15	1262.57	**987**	1080.19	60.17
	15	N/a	1296.94	1037	1104.02	62.83
TRENT	8	7	1363.37	**981**	1057.20	60.49
	15	5	1282.43	**951**	1043.64	64.95
	15	10	1284.36	**933**	1027.38	64.10
	15	15	1359.57	992	1069.84	57.87
	15	N/a	1259.89	985	1039.48	71.45

Table 5.23– Second-Order Statistics for Two-Trail Philosophy when Robust Elitist Settings are used for both trails and e_1 and e_2 are varied.

These results indicate that the use of a second population does bear some favourable influence on second-order solution quality. The bolded statistics are used when improvement is noted above the one-trail *Robust Elitist Settings* system. There is a reduction in first-order quality, but this is small. However, the use of two populations does have some drawbacks. Firstly, solutions constructed by the second population may be inferior to those deriving from the first population. In consequence, relatively poor solutions may be injected into the first trail from the second trail. Secondly, the increase in computational effort that is needed. Allowing two populations to perform construction automatically doubles the runtimes. Ant algorithms can be fairly computationally expensive on occasions and doubling this expense is unwelcome while considering the net second-order conflict gain.

5.11.2 Different Objectives

Here, we change focus slightly. The role of the second trail (refer to a specific fitness function) will be altered to become more similar to the work discussed by Doerner et al. (2001). The first trail will use the *CRF* and will be biased towards solving the first-order problem. Meanwhile, the second trail will utilize the second-order reward function (*SOF*) $\frac{\delta}{Sec\_Score}$, which attempts to construct timetables that are biased towards lower second-order scores.

5.11.2.1 Second-Order Function

Introducing an additional trail with a second-order biased reward function (SOF) will only mean benefit if the second-order scores of timetables constructed are lower than those previously experienced. By the nature of ants, the higher the trail level that is placed, the greater the probability to create a timetable with lower second-order conflict scores. Initially, we observe this additional benefit through experiments that use the reward function $\frac{\delta}{Sec\_Score}$. Only one trail is used here. Also note that no trail enhancement techniques have been incorporated and consequently, the results in Table 5.3 act as direct comparison to those presented below.

δ	HEC			EAR			TRENT		
	Av	*Best*	*Best 10*	*Av*	*Best*	*Best 10*	*Av*	*Best*	*Best 10*
1	1457.23	831	952.58	1704.75	1330	1519.18	1464.21	1090	1194.04
10	1308.49	774	1015.06	1741.29	1455	1571.03	1449.60	1117	1254.68
25	1449.44	862	969.00	1729.59	1405	1557.14	1450.03	1203	1233.42
50	1584.06	828	991.95	1761.47	1426	1575.23	1500.92	1132	1257.36
100	1416.13	819	969.20	1588.09	1280	1416.40	1514.83	1142	1264.92
250	1327.52	839	952.14	1701.15	1421	1542.40	1551.13	1267	1333.36
500	1256.82	751	904.90	1628.79	1308	1448.40	1444.08	1142	1206.80
1000	1349.89	758	1161.16	1687.47	1330	1520.58	1461.17	1146	1207.06
2500	1339.32	789	967.70	1636.87	1288	1424.88	1530.45	1125	1262.24
5000	1451.95	754	937.70	1824.31	1460	1637.43	1462.42	1106	1219.30

Table 5.24– Second-Order Statistics for SOF when One Trail is used.

We note here the reduction in average second-order scores (in comparison to the CRF system), in particular the differences in results with the HEC data set. Additionally, there seems an improvement in the 'best' timetable second-order scores. This seems more obvious with both HEC and TRENT. Noticeably though, the results appear rather insensitive to the weight δ placed on the reward function (observed earlier in Chapter).

However, despite the presence of no first-order criterion, the feasibility rate is acceptable (see Figure 5.35), although there is no pattern across the weights evaluated. Since the trails represent purely second-order feedback, the first-order constructional behaviour depends heavily on the single pass construction heuristic that is used.

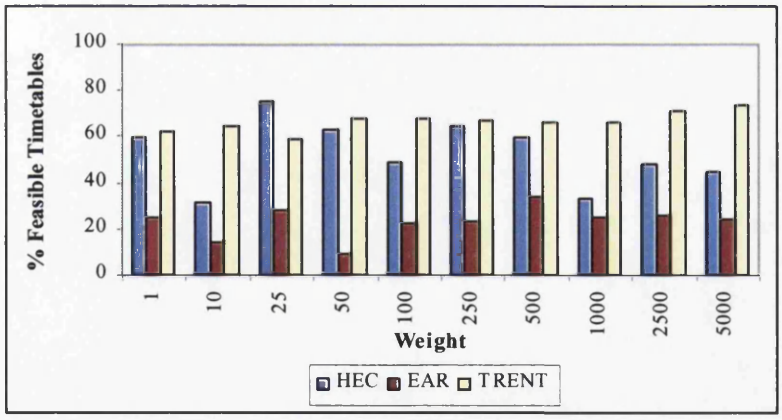


Figure 5.35 – Percentage of Feasible Timetables across all data sets for range of δ on SOF.

Let us gain some insight into the full benefit (with respect to second-order quality) this reward function could offer. Below are plots (Figures 5.36-5.38) of the average second-order scores for all timetables (irrespective of feasibility status) when using SOF and CRF. Each cycle observation point is averaged across five runs.

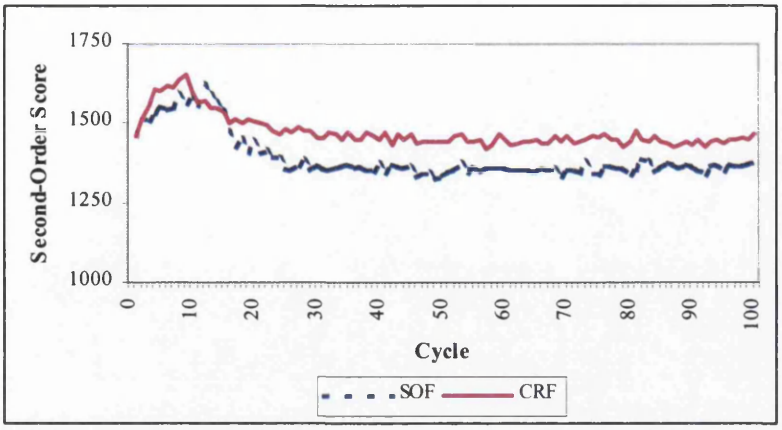


Figure 5.36 – Average Second-Order Score versus Cycle Plot comparing SOF and CRF for HEC.

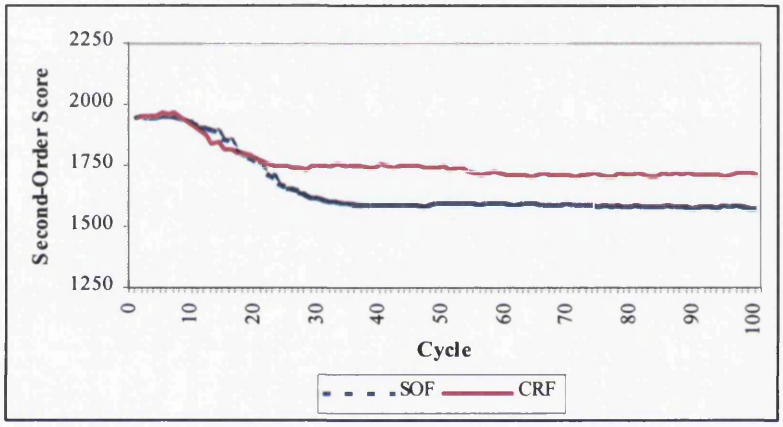


Figure 5.37 – Average Second-Order Score versus Cycle Plot comparing SOF and CRF for EAR.

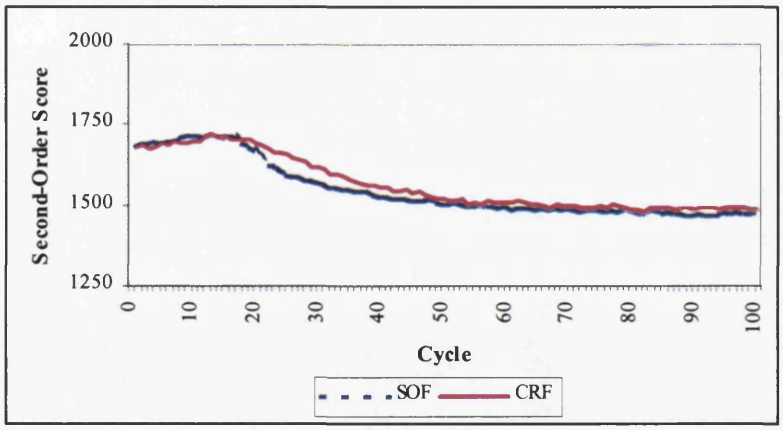


Figure 5.38– Average Second-Order Score versus Cycle Plot comparing SOF and CRF for TRENT.

There is a quality gap across the data sets. For each example, the use of *SOF* rather than *CRF* generates timetables of superior second-order performance. For HEC and EAR, the jump in quality is clearly evident. As improvement tails-off, there is a quality difference of

approximately *150 students*. A smaller difference exists with the TRENT data set. The results and illustrations presented above suggest that some exploitation of *SOF* is warranted.

5.11.2.2 CRF and SOF

We will look at using the *CRF* for the first population and the *SOF* for the second. A subset of solutions belonging to the second trail is used for extra replenishment for the first trail. However, no additional feedback is passed to the second trail since its aim is to optimise timetables with respect to second-order and consequently, any additional information could dilute the second-order knowledge gathered. The following set of Tables present the influence of the number of ants, a, that pass information from the second trail to the first and the weight of replenishment, σ.

	HEC			EAR			TRENT		
a	*Av*	*Best*	*Best 10*	*Av*	*Best*	*Best 10*	*Av*	*Best*	*Best 10*
1	1501.23	801	989.54	1754.30	1343	1500.12	1499.87	1154	1234.32
5	1475.07	861	939.44	1689.19	1231	1481.38	1497.90	1078	1219.20
10	1551.70	825	918.78	1712.95	1362	1488.16	1566.23	1196	1290.56
15	1486.17	779	932.86	1710.43	1316	1496.48	1504.73	1138	1233.86
20	1512.26	848	977.88	1633.43	1318	1413.03	1541.47	1209	1266.90
30	1605.23	807	995.24	1710.07	1294	1492.13	1448.83	1113	1191.42

Table 5.25– Second-Order Statistics across a ants when CRF and SOF are used for the 1st and 2nd trail respectively.

The results appear rather insensitive to the choice of a. However, based on previous findings, the number of exchange ants, a, is set at 15 for all data sets for the following set of experiments. We now vary the replenishment rate σ.

	HEC			EAR			TRENT		
σ	*Av*	*Best*	*Best 10*	*Av*	*Best*	*Best 10*	*Av*	*Best*	*Best 10*
5	1544.75	843	901.76	1656.48	1341	1517.00	1437.82	1105	1190.54
10	1295.14	754	910.34	1390.44	1232	1261.20	1405.56	1129	1231.02
15	1172.91	728	923.68	N/a			1427.99	1190	1284.00
20	1325.55	819	990.47				1639.34	1423	1510.70
30	N/a						N/a		

Table 5.26– Second-Order Statistics across intensification weight σ when CRF and SOF are used for the 1st and 2nd trail respectively.

Increasing the replenishment weight σ severely reduces the first-order capabilities of the first trail. When higher weighted second-order feedback is passed from the second trail, it

has the effect of creating an imbalance between first and second-order priorities for the first trail. Consequently, the first trail will struggle to construct feasible timetables and explains the 'blanks' in the above Table. The 'N/a' applies when not one feasible timetable has been observed. For higher settings of σ, the number of infeasible runs is high and the volume of feasible timetables is consequently low. The volume of feasible timetables is presented in Figure 5.39, which highlights the first-order problem. It appears that $\sigma=10$ is robust for all data sets.

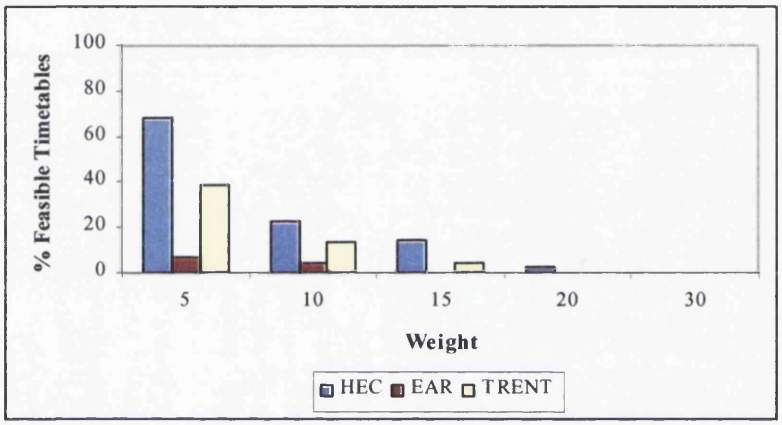


Figure 5.39 – Percentage of Feasible Timetables across Intensification Weight σ

As acknowledged above and consistently throughout this chapter, additional attention to the second-order problem reduces the first-order effectiveness of the search. Even though the overall task is to construct one timetable of perceived good second-order quality, it appears sensible here to readdress the balance between first and second order issues. Consequently, extra attention will be paid to the first-order problem. Section 5.5.3 showed that the 'elitist' intensification of trails, with appropriate parameter settings, had a dual influence – the improvement of second-order scores along with the maintenance of good levels of feasibility. The settings of a and σ remain ($a=15$, $\sigma=10$) for the 2nd trail and the number of elitist ants for the 1st trail is set at 15, but here we vary the intensification weight σ_1. The results are tabulated below.

σ_1	HEC			EAR			TRENT		
	Av	*Best*	*Best 10*	*Av*	*Best*	*Best 10*	*Av*	*Best*	*Best 10*
1	1172.91	728	923.68	1390.44	1232	1261.20	1465.56	1129	1230.02
5	1144.89	702	768.58	1327.10	1094	1142.28	1348.90	1019	1103.52
10	1162.81	673	746.68	1409.75	1094	1190.16	1339.53	1027	1090.42
15	1219.92	710	777.42	1479.24	1041	1221.10	1285.97	966	1041.78
20	1088.41	651	727.72	1476.88	1135	1259.30	1346.14	1000	1081.34
30	974.97	692	731.42	1331.94	1079	1158.64	1300.25	999	1084.24

Table 5.27– Second-Order Statistics across CRF Intensification Weight σ_1 when CRF and SOF are used for the 1st and 2nd trail respectively.

There is a marked improvement in solution quality through the use of elitism with the first trail. Encouragingly, the trade-off between first and second-order priorities is favourable. The following Figure, which illustrates the percentage of feasible timetables that are generated, provides evidence of the impact of σ_1. When no additional trail intensification ($\sigma_1=1$) is applied to the first trail, the percentage of feasible timetables is low (HEC – 7.03%, EAR – 1.90% and TRENT – 6.86%) but there is a marked increase in the feasible timetable count when σ_1 is set greater than 1. Second-order solution quality increases for $\sigma_1>1$ but, this is a consequence of increased feasibility.

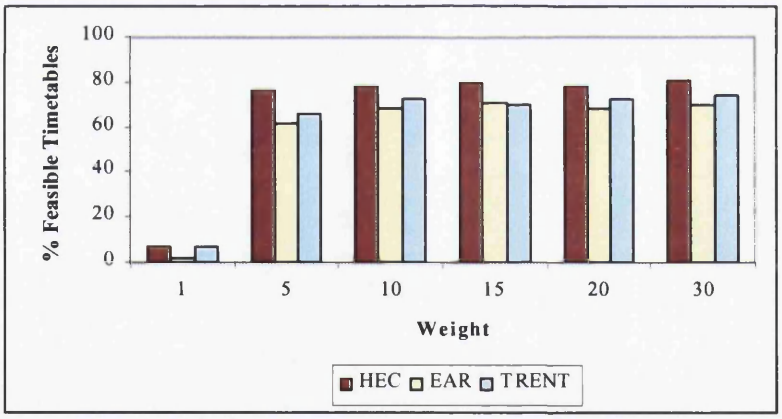


Figure 5.40 – Percentage of Feasible Timetables across Intensification Weight σ_1

It has been demonstrated that the use of a second-trail that attempts to optimise the second-order problem (irrespective of feasibility) does have the potential to generate better quality (with respect to second-order) solutions. Table 5.24 and Figures 5.36-5.38 have shown this. Figure 5.35 indicates that the feasibility rate is reasonable considering that no first-order feedback is passed to the trail. Although, the introduction of trail intensification reduces the first-order capabilities further (Figure 5.39). Overall, solution quality of the one trail system is generally better. Additionally, since the one-trail environment requires

lower computational effort, the reason for utilising two independent trails becomes obsolete.

5.12 One Population - Two Trails

In Section 5.11, we discussed the use of two populations that addressed different objectives. The second population injected information regarding the best solutions into the first population. This action aimed to diversify the search while attempting to exploit feedback that potentially could contribute towards better second-order scores. Also, it suggests another form of elitism since pairwise exams with relatively large accumulations of pheromone, with respect to the second trail, would be made available to ants belonging to the first trail and these accumulations will consequently have the potential to bias the search. In this section the two-trail philosophy remains but the focus changes.

The second trail will accumulate pheromone between exams that have been allocated to adjacent timeslots rather than between pairwise exam sets that have been scheduled in the same slot. The ambition is to recognise the exams that when scheduled in adjacent slots, contribute to 'better' second-order conflict scores.

5.12.1 Second-Trail Formulation

Let us define the first trail as F and the second as S. S is represented as an n by n matrix (n=number of exams) which is updated at predetermined times during the search. Given two exams e_r and e_t, the value S_{rt} in S is proportional to the quality of the timetables obtained by placing e_r and e_t in adjacent timeslots. Like F, the trail stored in S evaporates progressively as time goes by. The coefficient of evaporation is denoted by $(1-\varepsilon)$. Let $b_1,\ldots,b_{nants}$ be the timetables obtained at the end of the cycle and let $B_{rt} \subseteq \{b_1,\ldots,b_{nants}\}$ denote the subset of solutions in which e_r and e_t are allocated one timeslot apart. Let *fitness* be the defined fitness of the timetables used in b_d ($1 \le d \le nants$). The values S_{rt} in S are updated as follows

$$S_{rt} = \varepsilon \cdot S_{rt} + \sum_{b_d \in B_{rt}} fitness \quad (5.14)$$

Given a partial solution $s[k-1]=(e_1,...,e_q)$, with q referring to the number of exams in the partial solution, an unallocated exam e and a feasible timeslot t, the second-order trail factor $\sigma(s[k-1],e,t)$ is calculated as follows

$$\sigma(b[k-1],e,t)=\begin{cases} 1 & if\ (t-1)\ and\ (t+1)\ are\ empty \\ \dfrac{\sum_{x\in t-1} S_{xe}}{\psi_{t-1}}+\dfrac{\sum_{x\in t+1} S_{xe}}{\psi_{t+1}} & \end{cases} \quad (5.15)$$

where ψ_i represents the number of exams allocated to the i^{th} timeslot. As detailed in previous chapters, the solutions are built probabilistically on the basis of the knowledge drawn from previous solutions. The underlying concept remains the same here. The probability of allocating exam e to timeslot t (p_{et}) is defined as follows

$$p_{et}=\frac{[\tau(b[k-1],e,t)]^{\alpha}\cdot[\nu(b[k-1],e,t)]^{\beta}\cdot[\sigma(b[k-1],e,t)]^{\gamma}}{\sum_{r\in T_e}\{[\tau(b[k-1],e,r)]^{\alpha}\cdot[\nu(b[k-1],e,r)]^{\beta}\cdot[\sigma(b[k-1],e,r)]^{\gamma}\}} \quad (5.16)$$

In Section 5.4, the bias constant $[\eta(b[k-1],e,t)]$ was introduced and its involvement was aimed to bias exam insertions towards timeslots of lower second-order scores. One disadvantage to this technique rests with its utilisation. Using the *OddEven* approach to timeslot ordering in conjunction with RLF construction will only use this constant when the even numbered timeslots are being filled. A second drawback is the absence of a learning facility, which encourages greediness. Despite these pitfalls, results have shown that this bias constant did offer some benefit with respect to second-order scores. This leads one to incorporate a similar concept but introduce a learning facility in the form of a second-trail matrix. The role of the bias constant will be obsolete here and the trail S will be used instead.

The second trail factor (as defined in equation 5.15) is raised to the bias parameter γ and will be used in the following study.

5.12.2 Sensitivity of γ

The definition of *fitness* used here is the standard *CRF* expression. This reward expression is used for both trails. Additionally elitism is used and the relevant parameters are levied at the *Robust Elitist Settings*. The standard parameter settings of $\alpha=2$, $\beta=1$ and $\rho=0.5$ are used here.

In this section, we test the sensitivity of γ, which is the bias parameter associated with the second trail matrix. The results are as follows.

	γ	*Av*	*Best*	*Best 10*	*Worst 10*	Feasible
HEC	*0.5*	888.94	681	728.96	2351.76	83.12
	1	1032.17	637	720.32	2410.10	81.95
	2	1044.46	667	733.66	2448.78	80.63
EAR	*0.5*	1399.28	1087	1199.02	1968.50	72.20
	1	1350.86	1109	1172.30	1868.70	72.77
	2	1360.16	1085	1172.82	1903.52	75.93
TRENT	*0.5*	1360.93	1033	1102.44	1985.10	79.50
	1	1320.47	1019	1090.56	1908.00	78.67
	2	1320.88	1056	1085.44	1955.50	77.07

Table 5.28– Examining the Sensitivity of γ

The choice of γ seems unimportant to the overall success of the search. Subsequent experiments will use $\gamma=1$. The feasible timetable rate is favourable and while, the second-order performances are acceptable, the results do not compete with the solution quality associated with some previous studies in this chapter.

5.12.3 Different Objectives

In this section, we attempt to place extra emphasis on second-order. The second trail now refers to the *SOF* (with the first trail referring to the *CRF*) as the feedback mechanism. Elitism is used once more to emphasise relatively superior trails. The *Robust Elitist Settings* are used for the first trail while elitist related experiments are performed in relation to the second trail. The reward associated with each elitist solution belonging to the second trail is weighted by a constant σ. A range of σ and then the number of elitist ants, e, are tested and the statistics associated with a subset (involving parameter settings near the observed *Best*) can be viewed below. When σ is varied, e is set to 15.

HEC

The results for various σ are displayed in Table 5.29.

σ	Av	Best	Best 10
250	1015.03	773	800.52
500	1036.03	**643**	737.04
750	1007.61	680	769.40

Table 5.29– Influence of σ on second-order solution quality for HEC

Setting σ=*500* generates the best results and this setting is used to test the influence of varying the number of elitist ants. The results are presented in Table 5.30.

e	Av	Best	Best 10
3	991.38	734	803.84
5	986.38	**599**	728.68
10	978.92	635	745.30

Table 5.30– Influence of e on second-order solution quality for HEC

Reducing the number of elitist ants below 15 does improve solution quality marginally. The search process benefits from the reduced trail concentrates that result from fewer elitist ants.

EAR

Results, when σ is varied, are presented in Table 5.31.

σ	Av	Best
500	1370.60	1160
750	1332.24	**1061**
1000	1369.09	1076

Table 5.31 – Influence of σ on second-order solution quality for EAR

There is no emerging pattern as σ is varied but the best *Best* is observed at σ=*750* and this setting is used when *e* is varied.

e	Av	Best	Best 10
5	1375.33	1080	1199.78
10	1396.08	**969**	**1176.56**
15	1332.24	1061	1191.54

Table 5.32– Influence of e on second-order solution quality for EAR

The best *Best* observation of 969 is achieved when $e=10$.

TRENT

Results, when σ is varied, are presented in Table 5.33.

σ	Av	Best
500	1281.86	1013
750	1329.20	**1010**
1000	1359.45	1082

Table 5.33 – Influence of σ on second-order solution quality for TRENT

The best *Best* result is achieved when $\sigma=750$ and this setting is used when e is varied and those results are displayed in Table 5.34.

e	Av	Best
15	1329.20	1010
20	1289.57	**968**
25	1294.43	1001

Table 5.34 – Influence of e on second-order solution quality for TRENT

The influence of σ appears limited and some improvement can be achieved by resetting the number of elitist ants.

Overall, the use of elitism in conjunction with the multiplicative second-trail does offer benefit to solution quality. However, this influence is not very significant. Experiments associated with the results displayed (Tables 5.29-5.34) above have shown that appropriate settings of σ across the data sets are robust and optimum settings of e appear to increase as the size of problem does likewise.

In summary, it is demonstrated that the use of the second-trail does have the ability to construct better solutions than observed with the one-trail system, but not to a very significant level. Solution *Bests* have improved as follows. For HEC, the one-trail *Best* of 673 compares to the two-trail *Best* of 599, for EAR, 1037 drops to 969 and for TRENT, 985 is marginally poorer than 968. An intuitive reason for these (marginal) improvements is as follows:

At each decision stage, the first and second trail factors contribute to the construction of probabilities. Experiments were performed where the i^{th} ant in each cycle was monitored. All first and second trail factor scores were stored and when Pearson's correlation[2] was applied, it was found that the relationship between first and second trail factors exhibited positive correlation (0.67, 0.54, 0.51 for HEC, EAR and TRENT respectively)[3] and this association was deemed statistically significantly in each case. This observation may indicate why the search process benefited marginally from the inclusion of a second trail. Let us consider the array of first and second trail factor scores associated with feasible uncoloured exams at some exam-timeslot decision stage. In many cases, these scores are related with respect to rank but on rare occasions, a strong second trail factor score will lead to the insertion of an exam that otherwise would have a low probability of selection at that stage of the construction process.

It is noted here that experiments have been performed on the second trail evaporation factor. It has been found that this parameter plays no influence upon solution quality. However, this is no surprise due to the conclusions that were drawn regarding the trail evaporation parameter ρ in Chapter 3.

5.13 Conclusion

This chapter has focused on the second-order problem when scheduling examination timetables. It detailed a series of enhancement techniques that could be used to construct feasible timetables of better second-order quality. Initially, the reward function was altered to incorporate second-order characteristics of a timetable. Improvement was limited and consequently, it was shown that trail intensification was needed to enhance solution quality. Subsets of 'better' timetables were identified and extra trail was laid on these solutions. Basic static and dynamic intensification techniques were used. The Record-to-Record algorithm inspired the latter. These techniques showed hints of solution improvement. Consequently, more sophisticated techniques of trail intensification were applied - namely *Elitist Strategy*. The latter was attempted in a variety of ways. Focus was initially placed on the second-order fitness, but this was to the detriment of first-order

[2] Sample sizes for HEC, EAR and TRENT were large enough to allow use of parametric calculation.
[3] Correlation coefficient resides in interval [0,1]. Above Pearson's statistic deemed significant due to sample size used (18204 for HEC, 47601 for EAR and 62387 for TRENT).

quality. In consequence, the *Elitist Reward Function (ERF)*, which placed additional reward on both the first and second-order characteristics of the timetable, was used. Additional bias was then placed on the second-order component of the 'elite' timetables. Satisfactory universal settings (*Robust Elitist Settings*) were achieved for the *ERF*. However, it is not assumed that these can be applied generically. This method clearly showed that the ants communicated to enable the construction of feasible timetables with better second-order quality. However, elitist based methods can lead to higher trail intensities and limit the exploration of the ants. To counteract this, trail limits were imposed and superior results were achieved through the wider exploration performed. Also, additional improvement was gained through weighting the construction heuristic by the number of student clashes.

It was discussed that the framework of the DSatur heuristic has greater potential to manipulate timetables of improved solution quality than the RLF type heuristics. However, this approach was deemed as unsuitable due to the difficulty of generating feasible timetables.

In this chapter, a second population was introduced that searches the solution space according to the cost function that minimises second-order. It was demonstrated that marginally superior (than achieved through the *ERF* system) solutions were manageable by allowing two colonies to work in parallel and exchange information at specific times. However, the solution quality-computational effort trade-off is not favourable. The focus was then altered to use one population of ants but utilised a second trail that used a different update procedure, which allocated pheromone to pairwise exams belonging to adjacent timeslots. It is demonstrated that trail intensification is needed to generate relatively good solutions but these results are only marginally better than the one trail *ERF* based system.

This chapter has investigated methods of encouraging ant colony optimisation to produce good quality, feasible examination timetables by changes to the algorithm itself. However, other researchers (see Chapter 2 for examples) have found that ant colony optimisation has required additional assistance to produce extremely good results and the following chapter investigates such ideas.

Chapter 6

Improvement Strategies

6.1 Introduction

Chapter 5 introduced means to reduce the number of second-order conflicts present within a feasible timetable. A variety of methods were employed and many encouraged second-order scores lower than those timetables constructed under purely first-order conditions. However, literature that details the application of meta-heuristics to the examination-timetabling problem has shown the benefit of a hybrid or memetic approach – for example Burke (1995c), Thompson and Dowsland (1998). The meta-heuristics construct timetables with respect to first and second-order conflicts and some *Add-On* procedure is used as a repair facility to improve the fitness of a timetable with respect to some criterion e.g. minimizing second-order. Additionally, ant-based procedures for various problem types have also used hybrid methods, for example the Hybrid Ant System for the Quadratic Assignment Problem (HAS-QAP), Gambardella et al. (1999b), and for the Sequential Ordering Problem (HAS-SOP), Gambardella and Dorigo (2000). The uses of hybrid methods for meta-heuristic examination timetabling solution algorithms and for ant-based approaches for a variety of problem types strongly suggest that AS-EXAM will also benefit from the use of an *Add-On*. This chapter will discuss a collection of *Add-On* procedures. Firstly, a TSP based model will be used to reduce second-order conflict scores. The timeslots will be rearranged to an order that minimizes the back-to-back scores. Secondly, exam exchanges are performed through local search. Exams are allocated to alternative timeslots if a reduction in the second-order score will be observed in consequence. Finally, exam exchanges are made based on the well-respected graph theoretic Kempe Chain method. To symbolise the incorporation of an *Add-On*, the examination timetabling method discussed within this thesis will now become the Hybrid Ant System for the Examination Timetabling Problem (HAS-EXAM).

The structure within each distinct section is inspired by three different evolutionary strategies – Darwinian, Baldwinian and Lamarckian. In Darwinian evolution, learning does not play a significant role in the evolutionary process. Any traits learnt by the individuals of one generation are not passed onto the next. An advancement was first made by Jean Baptiste Lamarck, in 1815, when he claimed that all traits passed from one generation to

the next consist of both innate and acquired characteristics of the parents. James Baldwin, who introduced the *Baldwin Effect*, Baldwin (1896), took the middle ground between Darwin and Lamarck. His idea was that the fitness value of the improved individual could be transferred to the individual, without actually changing its genotype[4]. The link between these traditional evolutionary theories and evolutionary algorithms is a natural one. When using Darwinian theory, modifications are made to the solutions, however new characteristics and improved fitness values are not inherited by future constructed solutions. Conversely, when using Lamarckian theory, after modifications are made to solutions, both the new characteristics and bettered fitness values are passed on to future solutions. As stated above, Baldwinian theory takes the middle ground between Darwin and Lamarck. Here, modifications are still permitted but no solution characteristics are inherited by future constructions. However, the fitness value is associated with the original solution. In other words, Baldwinian strategy attempts to recognise constructions that offer the potential to produce better constructions after modification. The obvious application of these evolutionary theories is Genetic Algorithms (GA). Each time a chromosome is generated, an improvement strategy examines some nearby chromosomes and the best of these replaces (Lamarck) the original or transfers information to the original (Baldwin). Bull and Fogarty (1994) and Coyne and Paton (1994) both discussed the value of the principle of *talkback* (Lamarck) , also see Burke et al. (1995c) for a memetic algorithm for examination timetabling, which enables the recoding of the genotype, based on searches performed. Meanwhile, Julstrom (2000) interestingly compared all three theories when considering a genetic algorithm for the 4-cycle problem[5]. The work performed in that paper inspired the structure of this chapter. Lamarckian theory has also been used with ant algorithms, for example Gambardella and Dorigo (2000), but, in most cases, it has been performed implicitly. When Lamarckian theory has been used, the ants construct solutions and these solutions are modified according to the criterion used. Pheromone trail replenishment is executed with respect to the modified solution.

After each of the *Add-On* types have been discussed while compared across Darwinian, Baldwinian and Lamarckian evolutionary strategies, the trade-off between solution quality

[4] The internal genetic code carried by all living organisms. This information is passed from one generation to another.

[5] The objective of the 4-cycle problem is to observe the shortest collection of disjoint 4-cycles on a set of points in the plane (see Julstrom (2000) for details).

and computational effort is investigated and a direct comparison between the *Add-Ons* is performed. Towards the end of the chapter, the quality benefit gained through a selection of the methods introduced in both Chapters 5 and 6 are run on extra data sets. These results, along with some previous solutions, are assessed to deduce whether significant improvements in solution quality are gained. A preferred approach will be identified and a direct comparison of this will be made to solutions generated through a Simulated Annealing – Kempe hybrid strategy.

In this chapter, ant constructions will be based on the *Elitist Reward Function* system as described in Section 5.5.4.2.

The same five random number seeds are used for each set of experiments. Therefore, the same set of starting solutions (before Add-On is applied) between runs will be produced for the Darwinian study. However, starting solutions will differ between runs for the other evolutionary theories due to the nature of Baldwinian and Lamarckian theory.

6.2 Travelling Salesman Problem (TSP)

The Travelling Salesman Problem (TSP) has been defined in Appendix 1.2 but the definition will be revisited here. The TSP is the problem of a salesman who wants to find, starting at his home city, the shortest possible route through a set of customer cities and to return to his home city. More formally it can be represented by a complete weighted graph $G = (V,E)$ with V being the number of vertices, representing the cities and E being the set of edges that connect the vertices. Each edge is assigned a weight d_{ij}, which is the length of the edge $(i,j) \in E$, and represents the distance between cities i and j, with $i,j \in V$. The TSP is the problem of finding a minimal length Hamiltonian circuit of the graph, where a Hamiltonian circuit is a closed tour visiting each of the n cities once (where n represents the number of cities).

6.2.1 TSP for the Examination Timetabling Problem (TSP-ANT)

White and Chan (1979), Colijn and Layfield (1995) and Johnson (1990) have all have shown that the TSP can be applied to the examination timetabling problem. Once a feasible

timetable has been constructed, the towns represent the timeslots and the distance between the towns equates to the 'back-to-back' second-order clash scores between time periods. We rearrange the order of the timeslots such that the total second-order score is minimised. It should be noted here that, unlike the traditional TSP, the tour does not return to the start town and consequently, a path is constructed rather than a complete circuit. Additionally, no individual exam movements are allowed.

Rearranging the timeslots in such a manner makes intuitive sense but there is a drawback. The examination timetabling problem is highly constrained and any scheduler is usually required to accommodate pre-assigned and time-windowed exams. Such exams reduce the flexibility of the TSP approach. It only requires one exam to be fixed to a pre-assigned timeslot and the scheduler is forced to work around that timeslot. Other methods, which deal with exam rather than timeslot moves, have a higher potential to succeed. However, the TSP is suitable for data sets such as HEC, EAR and TRENT, which have no known side constraints.

The TSP is known to be NP-hard and as the number of towns increases the computational effort increases exponentially. Dorigo et al. (1991) proposed that an ant algorithm (Ant-Cycle) could be successfully applied to the TSP with runtimes that are minimal when compared to exact TSP method. The TSP-ANT philosophy will be used to tackle the examination scheduling problem. It could be argued that problems with small numbers of timeslots could be solved exactly, however such cases are rare and a heuristic TSP method is warranted for many examination problems.

Much attention will be paid to the TSP in this chapter. Since the TSP is the traditional testbed for ACO algorithms, it was felt warranted here to provide a more rounded investigation and almost treat the TSP as a second combinatorial optimization problem in its own right rather than solely an enhancement technique for the examination scheduling problem. Conclusions regarding the success of TSP as a repair mechanism will be drawn later after alternative approaches have been assessed.

It should be noted here that the TSP is suitable for the reduction of back-to-back conflicts. However, if a scheduler wishes to accommodate alternative student comfort criteria then an alternative cost function would have been used e.g. the cost function described in Carter et

al. (1996). If the timeslot structure differs (e.g. three slots per day with the aim of minimising back-to-backs within the same day) then a different method of enhancement may have to be used (see Chapter 2 for examples).

6.2.2 AS Algorithm for the TSP

An ant is an agent that moves from town to town on the TSP graph. When at town *t*, it determines town *t+1* using a probabilistic function combination of trail accumulated on the edges and of a heuristic value, which is a function of the length of the edges. Ants prefer shorter edges with a high concentration of pheromone trail. Initially, *n* ants (with *n* representing the number of ants per iteration) are placed uniformly on the towns and each ant constructs a tour that visits each town once. Each ant chooses the next town in its tour based on the following random proportional rule:

$$p_{ij} = \left\{ \frac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{j \in allowed} [\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}} \right\} \quad (6.1)$$

where *allowed* refers to the set of cities that can be visited after city *i*. This set must obviously not contain already visited cities and could exclude some not-yet visited cities at time *t* based on factors such as ordering constraints or candidate list structures, Dorigo et al. (1991).

The probability (p_{ij}) of an ant going from town i to town j is an amalgam of trail (τ_{ij}) and a visibility score (η_{ij}), which are biased by powers α and β respectively. The visibility score is a function of the distance (d) between one town and the next.

$$\eta_{ij} = \frac{1}{d_{ij}} \quad (6.2)$$

Each edge (i,j) on a constructed tour is updated according to the following reward function, which comprises the ratio between some quantity Q and the total distance of the tour D_t:

$$\Delta\tau_{ij} = \Delta\tau_{ij} + \frac{Q}{D_t} \quad (6.3)$$

This feedback is stored in a temporary trail matrix $\Delta\tau$. Each edge (i,j) is subjected to trail deposits and pheromone evaporation (ρ) at the end of each cycle and is modified according to the following equation:

$$\tau_{ij} = \rho \cdot \tau_{ij} + \Delta\tau_{ij} \quad (6.4)$$

6.2.3 Precursory Investigation

As has been commonplace with ACO, some preliminary investigation that deduces suitable parameter settings is required. The unknowns are:

1. The weight, Q, in the reward function.
2. The bias parameters α and β.
3. The evaporation rate (1-ρ).
4. The number of ants per cycle (N_A).
5. The number of cycles (N_C).

Only one parameter is varied at any given time and the remaining settings are as stated below (Table 6.1). Initial settings have been selected based on previous investigations and evidence from the literature. The weight, Q was suggested in Dorigo et al. (1991) along with the number of ants approximating the number of cities while it has been shown that the parameters α, β, and ρ encourage good algorithmic performance when set at 2, 1 and 0.5 respectively. Therefore, these values provide a sensible starting point for this investigation.

Parameter	Setting
Q	100
α	2
β	1
ρ	0.5
Ants per cycle	20
Cycles	50

Table 6.1 Fixed experimental parameter settings for TSP

After the construction of a solution by an ant, the ant-based TSP procedure improves the solution further. The *Best* solution observed during the TSP acts as the final solution. Statistics that are presented consider only the final solutions. Typically, *Av* relates to the average final solution across all the ants that have been improved via the TSP.

6.2.3.1 Weight Q

Table 6.2 presents the results of TSP related experiments for varied reward function weight Q.

	HEC			EAR			TRENT		
Q	*Av*	*Best*	*Best 10*	*Av*	*Best*	*Best 10*	*Av*	*Best*	*Best 10*
1	967.43	694	761.66	1567.97	1143	1263.78	1464.34	1219	1271.46
100	720.62	553	595.32	1476.94	913	1000.94	1331.70	1066	1115.44
500	765.33	549	615.60	1423.89	952	1084.06	1341.19	1067	1121.98
1000	765.74	561	624.94	1432.08	979	1110.38	1348.00	1058	1128.16
5000	780.87	552	632.72	1419.93	984	1090.97	1355.39	1074	1133.52

Table 6.2 Statistics describing the influence of reward function weight Q in TSP.

Robust solutions are produced for $Q \geq 100$, with lower Q generating inferior solutions. This is true for each of the data sets. It seems safe to conclude that $Q=100$ is an appropriate setting. Such experiments were presented in Dorigo et al. (1991) for $Q \in \{1, 100, 10000\}$ when using the Ant-Cycle algorithm. Favourable conclusions were made when $Q=100$.

6.2.3.2 Bias Parameters

We vary the bias parameters α (trail) and β (desirability) and the results are as presented in Tables 6.3-6.5 for HEC, EAR and TRENT respectively:

HEC

		β			
α		1	2	3	4
1	*Av*	*707.89*	*707.55*	1023.23	1054.37
	Best	*538*	*546*	715	754
	Best 10	*586.4*	*584.44*	765.66	815.94
	Worst 10	*853.08*	*887.36*	1283.7	1275.36
2	*Av*	*720.62*	*707.12*	942.47	1038.32
	Best	*553*	*543*	555	709
	Best 10	*595.32*	*578.56*	664.92	798.18
	Worst 10	*891.10*	*889.02*	1285.14	1274.94
3	*Av*	759.2	740.08	827.13	1019.44
	Best	554	551	575	685
	Best 10	603.88	590.04	616.7	763.26
	Worst 10	983.82	970.4	1210.8	1283.3
4	*Av*	783.6	765.06	829.19	959.88
	Best	560	551	588	587
	Best 10	610.4	597.32	630.44	665.32
	Worst 10	1032.3	1036.72	1152.4	1289.88

Table 6.3 – Statistics describing the influence of bias parameters α and β for HEC

EAR

		β			
α		1	2	3	4
1	*Av*	*1445.52*	*1505.07*	1653.12	1627.35
	Best	*957*	*1016*	1225	1224
	Best 10	*1132.02*	*1170.78*	1374.36	1349.06
	Worst 10	*1701.26*	*1799.96*	1909.30	1874.72
2	*Av*	*1476.94*	*1420.41*	1649.65	1642.24
	Best	*913*	*954*	1156	1214
	Best 10	*1000.94*	*1071.90*	1344.16	1360.95
	Worst 10	*1712.48*	*1776.50*	1936.5	1894.4
3	*Av*	1552.96	1537.27	1643.47	1634.99
	Best	989	1000	1123	1173
	Best 10	1130.68	1161.14	1264.10	1337.96
	Worst 10	1869.70	1880.44	1949.98	1899.18
4	*Av*	1548.87	1600.73	1641.88	1641.34
	Best	991	1009	1060	1259
	Best 10	1173.32	1211.64	1257.68	1403.30
	Worst 10	1875.87	1918.62	1943.62	1929.78

Table 6.4 – Statistics describing the influence of bias parameters α and β for EAR in TSP.

TRENT

α		β 1	2	3	4
1	*Av*	*1377.65*	*1371.71*	1474.55	1469.65
	Best	*1089*	*1122*	1205	1225
	Best 10	*1131.20*	*1171.14*	1272.94	1268.72
	Worst 10	*1500.56*	*1606.96*	1721.12	1700.76
2	*Av*	*1331.70*	*1344.44*	1489.53	1483.36
	Best	*1066*	*1068*	1191	1209
	Best 10	*1115.44*	*1120.04*	1263.74	1261.14
	Worst 10	*1565.14*	*1068*	1191	1209
3	*Av*	1407.73	1406.08	1406.07	1495.36
	Best	1115	1112	1095	1216
	Best 10	1165.64	1148.84	1148	1256.90
	Worst 10	1649.80	1675.42	1675.40	1728.86
4	*Av*	1436.89	1438.84	1460.61	1499.61
	Best	1140	1110	1104	1177
	Best 10	1189.16	1171.36	1187.86	1265.56
	Worst 10	1681.32	1707.00	1739.28	1739.16

Table 6.5 – Statistics describing the influence of bias parameters α and β for TRENT in TSP.

The results generated by the parameter constraints $\alpha < 2$ and $\beta < 2$ (represented by italics in Table 6.3-6.5) are superior to other bias settings. The quality disparity for α and β combinations when $\alpha \leq 2$ and $\beta \leq 2$ is minimal. This contrasts with the results achieved for the parallel study for the ANTCOL algorithm in Chapter 3, where it was shown that there is a significant jump in solution quality between $\alpha=1$ and $\alpha=2$. This can be attributed to the difficulty of the examination scheduling problem, which requires additional emphasis to be placed on the trail factor.

Chapter 3 demonstrated that raising random-proportional rule components (i.e. trail and heuristic) to a power greater than 1 has a detrimental effect on runtimes. Consequently, justified parameter settings here are $\alpha=1$ and $\beta=1$.

6.2.3.3 Evaporation Rate

The TSP trail not only accumulates information during the search process but loses some exploration knowledge as well. This allows the ants to be more focused on the recently accumulated information and dilutes the influence of poorer solutions found earlier in the search. Both Dorigo et al. (1991), for the Ant-Cycle algorithm, and Costa and Hertz

(1997), for the ANTCOL algorithm, suggest a setting of 0.5. Experiments have been performed where the evaporation rate ($1-\rho$) has been varied and its influence has been assessed.

	HEC			EAR			TRENT		
ρ	*Av*	*Best*	*Best 10*	*Av*	*Best*	*Best 10*	*Av*	*Best*	*Best 10*
0.1	758.05	552	607.82	1418.46	1038	1090.58	1366.08	1104	1144.98
0.3	738.62	559	602.50	1508.57	1024	1149.80	1357.93	1089	1137.84
0.5	765.74	561	624.94	1432.08	979	1110.38	1348.00	1058	1128.16
0.7	750.85	552	609.04	1463.44	1001	1122.20	1337.60	1076	1123.16
0.9	716.77	541	582.88	1422.04	990	1106.62	1325.81	1047	1119.32
0.99	755.08	557	620.54	1412.61	982	1110.60	1323.17	1067	1124.60

Table 6.6 – Statistics describing the influence of evaporation rate ρ in TSP.

Solutions seem a little insensitive to the setting of ρ. The only set of results that suggests ρ holds any influence is with the TRENT data set. As ρ increases, the second-order statistics lowers. Notably, using $\rho=0.9$ generates the best observed results. However, this evidence does not warrant a change of ρ from its original setting of 0.5. Within this thesis, work in Chapter 3 indicated that $\rho=0.5$ is a suitable choice. Since this version of the TSP is using a global updating rule and a cyclic framework, like the work detailed in Chapter 3, this adds extra weight to the argument that $\rho=0.5$ is appropriate.

6.2.3.4 Ants per Cycle

Dorigo et al. (1991) and Colorni et al. (1996) suggested that the number of ants should be set to the number of cities. This is a suitable rule that can be used when no time is available for some precursory investigation, which is a disadvantage of the use of ants. However, Table 6.7 presents the solutions that suggest that increasing the number of ants per cycle above the number of cities improves solution quality. The number of cities is 18, 24 and 23 for HEC, EAR and TRENT respectively.

	HEC			EAR			TRENT		
N_A	*Av*	*Best*	*Best 10*	*Av*	*Best*	*Best 10*	*Av*	*Best*	*Best 10*
1	1028.31	698	754.16	1698.32	1235	1355.84	1544.82	1202	1531.34
5	842.07	605	655.22	1574.18	1098	1212.08	1451.96	1133	1192.22
10	806.60	564	631.18	1473.16	1020	1126.98	1411.32	1102	1171.34
20	765.74	561	624.94	1432.08	979	1110.38	1348.00	1058	1128.16
30	712.23	522	597.32	1372.67	971	1076.28	1285.32	1001	1087.94
50	713.02	525	598.60	1365.25	965	1063.85	1298.60	1023	1096.08
100	716.77	559	598.30	1351.16	962	1045.54	1265.83	998	1083.08

Table 6.7 – Statistics describing the influence of the number of ants per cycle in TSP.

A robust setting of the number of ants per cycle has been selected as 30. The solution improvement appears to dampen as N_A increases. Even though the aim is to produce a timetable of good second-order quality, the TSP is only being used as an improvement strategy and consequently, the number of ants per cycle will be limited to make runtimes more attractive. Computational effort will be discussed in greater detail in Sections 6.2.9 and 6.2.10 but it should be noted that increasing the number of ants per cycle influences runtimes linearly. For example, 50 ants per cycle will require twice the runtime that 25 ants per cycle will need. This provides evidence of the benefit of reducing the number of ants per cycle.

6.2.3.5 Number of Cycles

The average latest cycle that a new best solution is found is tabulated:

Data Set	Average Latest Cycle
HEC	22.34
EAR	28.71
TRENT	23.65

Table 6.8 – Statistics describing the average latest cycle that a new best solution was found in TSP.

Observing that, on average, the final new bests are found before the *30^{th} cycle* and minimal improvement is achieved beyond this cycle number suggest that the number of cycles used in future experiments could be reduced from *50*. This will have the benefit of reducing computational effort while not sacrificing solution quality.

6.2.3.6 Parameter settings

Based on the findings presented above, suitable parameters settings are as follows:

Parameter	Setting
Q	100
α	1
β	1
ρ	0.5
Ants per cycle	30
Cycles	30

Table 6.9 – Final parameter TSP settings

6.2.4 Enhancing Solution Quality

Much work has been detailed regarding trail intensification strategies in the previous chapter. It was shown that allowing 'elite' ants to deposit extra levels of pheromone encouraged solutions of better second-order quality. This observation coupled with the benefit that elitism brings to the traditional TSP, Dorigo et al. (1991), suggests that the application of elitism to the TSP algorithm is a natural avenue of investigation. Applying elitism requires the parameterisation of some unknowns:

1. The number of elitist ants (e)
2. The weight placed on elitist trails (σ)

Like Section 5.5, some preliminary investigation, which determines the appropriate settings for these unknowns, is needed. Each parameter is varied independently and the other parameter is fixed as follows:

Parameter	Setting
e	3
σ	15

Table 6.10 - Fixed experimental parameter settings for Elitist TSP

The parameter e has been chosen based on the *10% rule* discussed in Section 5.5. The number of TSP ants per cycle has been set at *30*, which explains $e=3$. Meanwhile, the weight parameter σ is fixed at *15* due to the investigation presented in Section 5.5.

Influence of *e*

The influence of a range of *e* has been investigated. A copy of the *e* best solutions are stored and additional replenishment is passed to the trail according to these 'elite' solutions. The results are presented in Table 6.11.

	HEC			EAR			TRENT		
e	*Av*	*Best*	*Best 10*	*Av*	*Best*	*Best 10*	*Av*	*Best*	*Best 10*
1	720.86	544	599.08	1554.73	1041	1214.20	1283.96	1022	1068.44
2	703.27	546	594.54	1459.83	1053	1137.83	1228.81	985	1044.10
3	696.44	541	590.62	1409.59	1043	1125.53	1204.51	957	1023.12
5	687.96	534	587.64	1365.56	1008	1102.23	1182.64	945	1009.42
10	682.29	537	583.10	1344.03	1034	1096.13	1169.30	959	1007.36
15	682.08	534	582.92	1347.92	1024	1091.80	1168.69	960	1004.58
20	681.81	535	583.74	1357.71	1036	1098.90	1171.95	956	1006.18
25	682.47	534	582.18	1335.99	1000	1109.65	1322.36	1075	1145.98
30	763.34	554	628.66	1338.23	1000	1094.60	1386.45	1098	1171.64

Table 6.11 – Statistics describing the influence of the number of elitist ants e in TSP.

Across the data sets, restricting the number of elitist ants appears wise. Higher *e* generates poorer solutions for HEC and TRENT. Too many elite ants place higher concentrates of pheromone on a wider selection of edges. In consequence, the search becomes less focused on better solutions. Limiting the number of elitist ants to 5 across the data sets is suggested because the *Best* results are of high quality across all data sets.

Influence of σ

The influence of the trail intensification weight σ has been investigated. The trails associated with each elite solution are magnified by some constant to equip them with more influence. A similar study has been detailed in Section 5.5 and increasing σ improves solution quality.

	HEC			EAR			TRENT		
σ	*Av*	*Best*	*Best 10*	*Av*	*Best*	*Best 10*	*Av*	*Best*	*Best 10*
1	804.52	543	617.02	1678.17	1330	1428.83	1398.63	1083	1162.52
5	708.91	534	593.40	1635.75	1194	1306.85	1315.04	1009	1075.90
10	698.30	541	590.18	1505.59	1053	1161.38	1233.78	1004	1038.48
15	696.44	541	590.62	1409.59	1043	1125.53	1204.51	957	1023.12
20	695.46	546	592.32	1367.13	1023	1104.45	1193.09	986	1020.62
30	681.47	537	581.88	1322.99	1021	1084.75	1160.33	947	1002.36

Table 6.12 – Statistics describing the influence of trail intensification weight σ

The results displayed in Table 6.12 provide evidence that increasing the trail weight improves solution quality. As σ reaches *15*, its marginal influence reduces. Since higher σ could encourage stagnation, σ will be set to *15*.

6.2.5 Best Possible Search Conditions

The ant system has incorporated the elitist strategy during the construction phase and trail intensification has also been employed during the *Add-On* phase to encourage tours of smaller length, thus producing timeslot sequences of smaller second-order quality.

Data Set	Av	Best	Best 10	Worst 10
HEC	647.39	512	558.06	793.18
EAR	1130.76	842	939.20	1413.02
TRENT	1135.61	930	971.38	1354.84

Table 6.13 – Statistics associated with the best search conditions

Table 6.13 illustrates the important role that the ants can play if the searches are efficient and biased towards superior areas of the solution space. Let us make direct comparison to the results presented in Tables 6.3-6.5. These data correspond to TSP improvements made to solutions originally constructed when biased only to first-order conditions. The statistics presented in Table 6.13 are clearly superior and emphasise the importance of good ant-based solutions.

6.2.6 Heuristic Bias Only

The analogy of ants is based on the pheromone deposits, which represents solution fitness. Running the algorithm without these pheromone trails will provide an indication of the additional search potency that such 'bias' provides. Such runs will mimic the structure of the ANT-TSP and the selection of the next city in the tour will again be random proportional but now will be biased only according to the visibility score i.e. distance between two cities.

Data Set	Av	Best	Best 10	Worst 10
HEC	854.35 (8.69)	620 (7.88)	696.64	1009.16
EAR	1199.04 (7.54)	919 (11.38)	987.90	1520.20
TRENT	1184.42 (5.99)	963 (2.23)	1005.34	1489.31

Table 6.14 – Statistics describing the heuristic bias only conditions

The removal of pheromone has a detrimental effect on solution quality. Direct comparison can be made with Table 6.13. However, the use of a heuristic only system does yield solutions notable superior than the *Elitist Reward Function* system in Section 5.5.3.2 (See Table 5.16). The percentage improvement is represented by the terms in parentheses and these values indicate that the heuristic only procedure does provide overall benefit to solution quality.

6.2.7 Ant synergy

Figures 6.1-6.3 illustrate the cyclic best second-order scores averaged over all feasible timetables during the TSP improvement phase. Comparison is made between the search that uses both trail and desirability (*with pheromone*) and the search that utilizes only the desirability function (*without pheromone*). The plots represent one complete run with identical starting solutions being used (derived from same random number seeds).

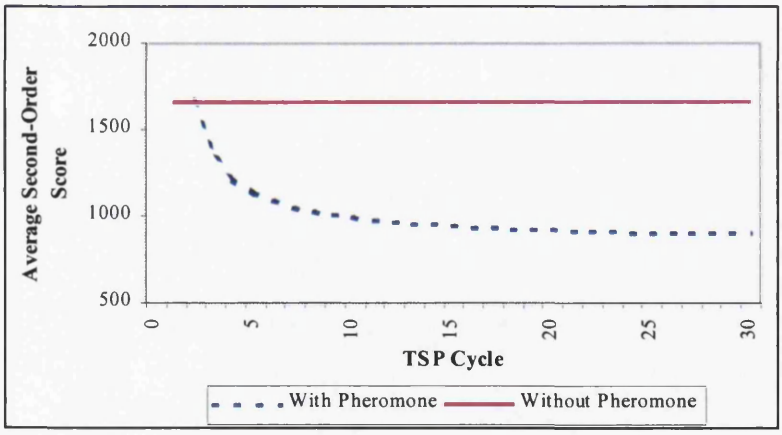


Figure 6.1 – Average Second-Order Scores for both with and without pheromone conditions for HEC

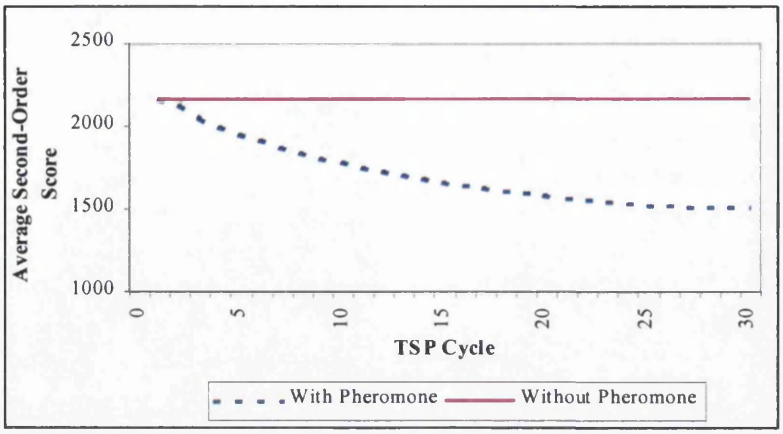


Figure 6.2 – Average Second-Order Scores for both with and without pheromone conditions for EAR

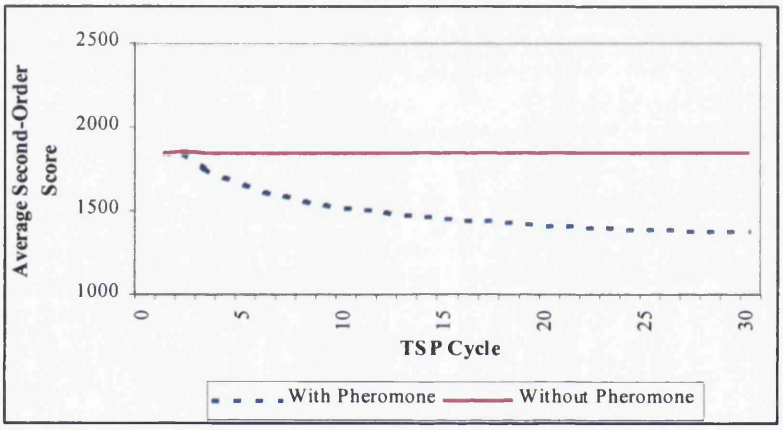


Figure 6.3 – Average Second-Order Scores for both with and without pheromone conditions for TRENT

The *with pheromone* line represents the benefit of ant synergy. The negative gradient signifies the improved averaged second-order score as the search matures. The zero gradient *without pheromone* plot further highlights the improvement in solution quality. Table 6.14 suggests that acceptable results are possible with heuristic only bias. The results presented in Table 6.14 are clearly superior to those displayed in Table 5.16.

6.2.8 Relationship between Before and After TSP scores

The underlying aim is to generate one feasible exam timetable of 'good' second-order quality. This suggests that the application of the TSP improvement approach could be refocused to a subset of timetables rather than, as to this juncture, each feasible timetable. Reducing the volume of timetables that are subject to TSP improvement will certainly reduce the computational effort required. A criterion that identifies a suitable subset of

timetables is investigated here. Therefore, it is interesting to determine whether a meaningful relationship between *Before TSP* (the ant-based solution before TSP is used to improve quality) and *After TSP* (best solution after TSP is used to improve quality) scores exists. To minimise the volume of data and eliminate noise elements the *Before TSP* data is ordered and assigned to intervals of width 100. Within each interval, these data are averaged along with the associated *After TSP* data. The following set of Figures illustrates the *Before TSP* and *After TSP* relationship. Each Figure depicts a positive trend, which suggests that lower *Before TSP* scores encourages lower *After TSP* scores. However, the possibility of a timetable with a high *Before TSP* score becoming a modified timetable with a 'relatively' low score does exist, although Figures 6.4-6.6 suggest that such an observation is rare. Some timetables are constructed with a low number of 'high conflict' pairwise exam sets that are scheduled one timeslot apart and these significantly contribute towards a 'poor' score.

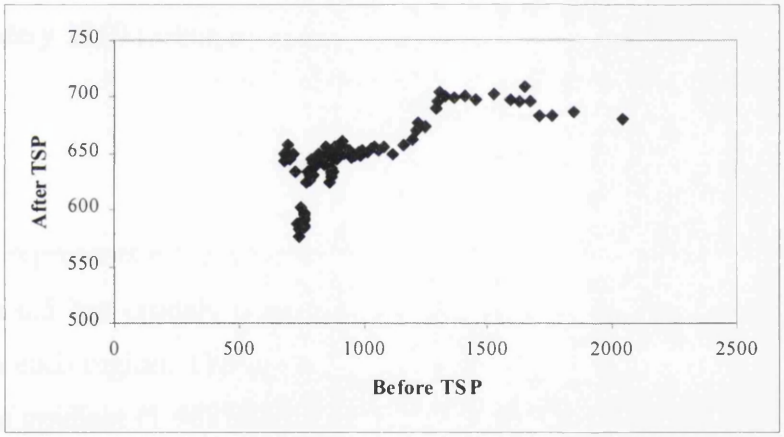


Figure 6.4 – Before TSP and After TSP relationship for HEC

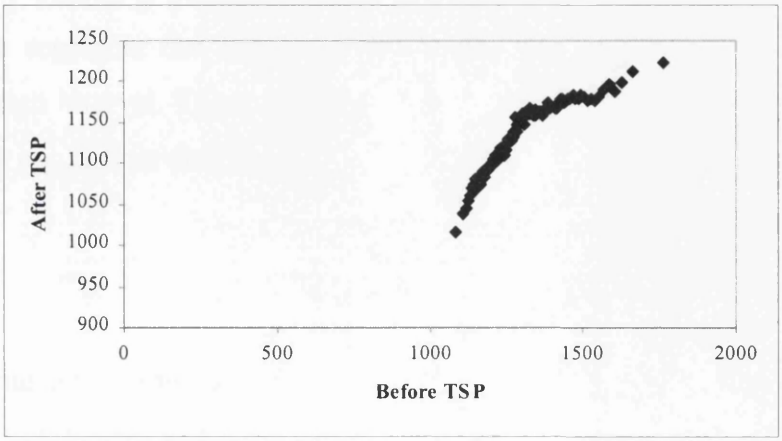


Figure 6.5 – Before TSP and After TSP relationship for EAR

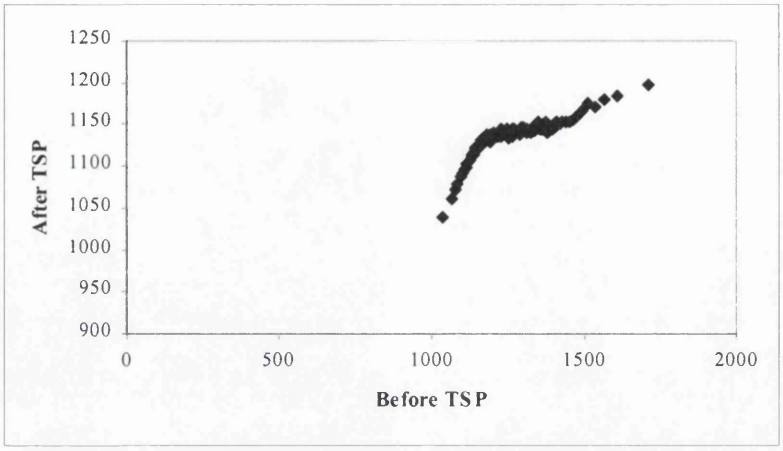


Figure 6.6 – Before TSP and After TSP relationship for TRENT

HEC

The scatterplot for HEC depicts a trend that is furthest removed from linear than any other data set. Generally, the trend is positive and the gradient dampens for *Before TSP* greater than approximately 1350 instances of students sitting back to back examinations.

EAR

The scatterplot represents a monotonically increasing relationship between the variables of interest. Figure 6.5 has crudely been split into two distinct regions and linear lines have been applied to each region. The gradient (6.87) in the *Before TSP* interval [1085,1300] is steeper than the gradient (1.56) belonging to the interval [1300,1767], therefore indicating the greater scale of improvement that belongs to the latter interval as the *Before TSP* drops. Therefore, if we wished to model this relationship, piecewise linear could be used on an ad-hoc basis to segregate the *Before TSP* into a series of intervals and conduct linear regression on each interval. This method would provide the marginal benefit that one less *Before TSP* unit could have on the *After TSP* score.

TRENT

The TRENT data set exhibits a similar trend to EAR - a rapid gradient for lower *Before TSP* scores (approximately less than 1200) and then the gradient lowers past this mark. The gradient appears smaller than experienced with EAR.

Attempts have been made to model and quantify the relationship between *Before TSP* and *After TSP* scores. Regression models that attempt to predict *After TSP* given *Before TSP* have been produced. The superior model type, due to the explained variation, is linear (see Appendix 6.1). However, this model, along with the other tried model types, fail the regression assumptions such as errors being normally distributed and consequently, if extensive use of these models were made then the conclusions drawn could prove unreliable. Not conforming to such assumptions may not be too surprising given the nature of the data set. The *Before TSP* and *After TSP* scores cannot be classified as a random sample as is required with statistical modeling. Members of the data set are biased according to the fitness criterion used by the ants during the searches. Towards the latter stages of the search ants produce a number of timetables of biased 'good' quality. Consequently, the data set is clearly biased. However, devising models that determine the relationship between *Before TSP* and *After TSP* scores would be useful if a generalised model type for any data set was possible – to quantify relationships. However, scale disparities between data sets suggest such a model will be impossible to create and individual models will have to suffice.

It was stated above that the regression models generated for each of the data sets did not observe the regression validity assumptions. Additionally, the distribution of second-order scores within each data set is clearly not Normal. Histograms and the Kolmogorov-Smirnov Test produced for both variables across the data sets prove this comprehensively. Given such observations, correlation tests will be non-parametric. Spearman's Rank will be used to quantify the strength of association between *Before TSP* and *After TSP* variables. The correlation coefficients are as follows.

Data Set	Spearman's Rank Correlation Coefficient
HEC	0.80
EAR	0.99
TRENT	0.95

Table 6.15– Spearman's Rank Correlation Coefficient quantifying Before TSP and After TSP relationship

All correlations are significant at the 0.01% level and indicate that lower *Before TSP* scores lead to better *After TSP* scores.

The evidence presented indicates that the relationship between *Before TSP* and *After TSP* could be exploited to reduce the number of timetables that are improved, thus saving computational effort. This study is detailed in Section 6.2.9.

An interesting relationship is *Before TSP* versus *Savings*, where *Savings* relates to the number of back-to-back clashes eliminated in the timetable after the application of TSP. As above, the *Before TSP* data is ordered and assigned to intervals of width 100. These data are averaged along with the associated 100 *Savings* data points. Plots for each of the data sets are presented below:

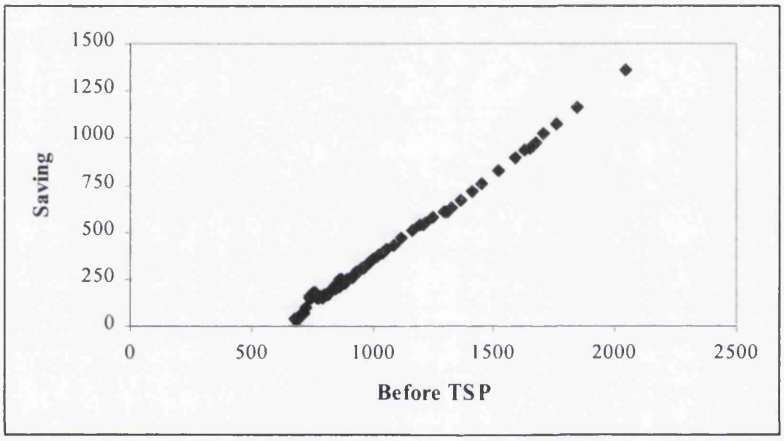


Figure 6.7 – Before TSP and second-order conflicts savings relationship for HEC

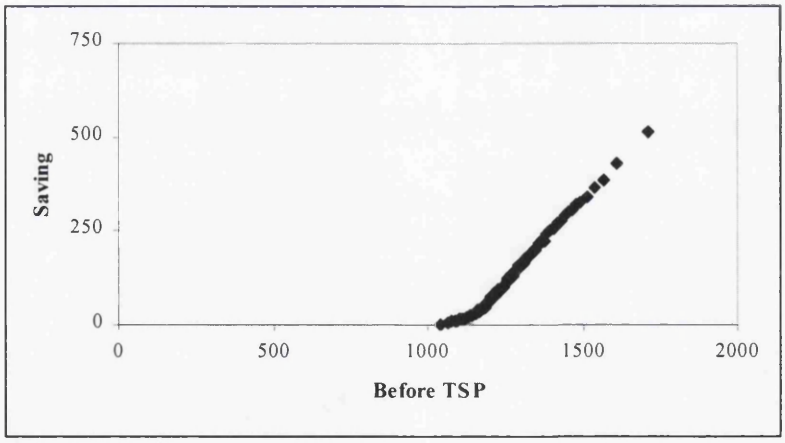


Figure 6.8 – Before TSP and second-order conflicts savings relationship for EAR

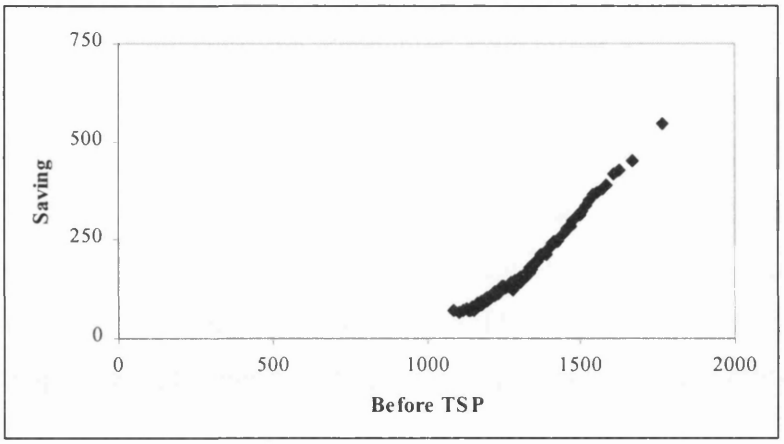


Figure 6.9 – Before TSP and second-order conflicts savings relationship for TRENT

Figures 6.7-6.9 depict a greater linear association than presented in Figures 6.4-6.6. Figures 6.7-6.9 all illustrate a striking positive relationship. For relatively low *Before TSP* scores, for each of the data sets, the marginal improvement slows down and does not appear to conform to a linear association. However, for larger *Before TSP* scores, the relationship appears to be almost perfectly linear. Through applying simple linear regression, we can see that the positive linear trend for each data set is statistically significant. The explained variation statistics, R^2, for each model is deemed as statistically significant at the *1% level of significance*. However, the residuals validity checks strongly suggest that these regression models cannot be accepted and alternative modeling strategies are recommended. Please refer to Appendix 6.1 for details on the regression model. As above, we turn to non-parametrics in attempt to quantify the strength of these associations. Table 6.16 presents the Spearman's Rank Correlation Coefficients for each data set.

Data Set	Spearman's Rank Correlation Coefficient
HEC	1
EAR	1
TRENT	0.99

Table 6.16 – Spearman's Rank Correlation Coefficient quantifying Before TSP conflicts savings relationship

All data sets have strongly significant relationships between the *Before TSP* and *Savings* variables, accepted at *0.001% level of significance.*

To conclude this section, evidence suggests that applying the TSP algorithm as an improvement strategy improves the acceptability of a timetable dramatically. Larger back-

to-back reductions originate from the timetables with larger *Before TSP* scores and the marginal improvement can almost be classified as linear. However, even though relatively poor *Before TSP* timetables can be transformed into relatively good *After TSP* timetables, the most suitable *After TSP* timetables are those that had superior *Before TSP* scores. However, Figures 6.7-6.9 indicate that improvement is limited for very good *Before TSP* timetables, thus almost removing the need of the TSP improvement technique for such timetables, but emphasizing the benefit of very good ant-based (*Before TSP*) solutions.

6.2.9 Quality-Time Trade-Off

Evidence has shown that better *After TSP* timetables originate from superior *Before TSP* timetables. Since the aim of a scheduler is to generate one feasible timetable of acceptable quality, it seems excessive to apply the TSP improvement strategy to all constructed feasible timetables. In this section, we will assess the trade off between quality and computational effort. The *Before TSP* timetables are ranked according to second-order fitness and the top t timetables are improved. The value of t varies according to the number of eligible timetables. If the *Before TSP* score is lower than a deviation above the run-minimum (the current best solution within each independent run) then TSP is applied such that

If Before TSP < Run-Minimum*(1+δ) {0≤δ<1} Then
 Improve via TSP
End If

Table 6.17 displays the solutions achievable given the deviation level used. The statistics have been averaged across five independent runs. Results regarding each individual run can be viewed in Appendix 6.2.

A range of deviations has been investigated and a selection is presented. The results are directly comparable against all other displayed deviations. Additionally, the percentage difference between the observed *Best* and *Time* of each deviation used and *infinite* deviation (no limit) are recorded.

	Deviation	Average	Best	% from Best	% fromTime
HEC	*Infinite*	647.39	512	0.00	0
	0	647.71	561	9.57	85.49
	1	633.88	526	2.73	82.02
	3	631.23	526	2.73	75.18
	5	634.76	526	2.73	64.17
	10	636.97	512	0.00	50.60
	20	641.47	512	0.00	31.59
	30	642.51	512	0.00	22.63
	50	644.22	512	0.00	15.87
EAR	*Infinite*	1130.75	842	0.00	0
	0	1085.70	895	6.29	64.17
	1	1086.60	895	6.29	63.98
	3	1084.82	868	3.09	63.01
	5	1089.18	868	3.09	60.32
	10	1104.63	852	1.19	45.00
	20	1123.62	842	0.00	15.47
	30	1129.10	842	0.00	4.24
	50	1130.68	842	0.00	0.08
TRENT	*Infinite*	1135.61	930	0.00	0
	0	1109.22	986	6.02	55.21
	1	1110.02	986	6.02	55.06
	3	1101.22	969	4.19	54.41
	5	1101.59	957	2.90	53.02
	10	1114.14	957	2.90	44.42
	20	1130.89	953	2.47	20.49
	30	1134.83	942	1.29	6.39
	50	1135.57	942	1.29	0.26

Table 6.17 – Second-Order After-TSP statistics for range of deviation scores

Table 6.17 suggests that much computational effort is wasted if attempts are made to improve every timetable. Acceptable timetables are possible for low deviations and since the number of modified timetables is reduced, runtimes are cut greatly. Therefore, the quality-time trade-off is encouragingly one-sided and this is observed across all runs for all data sets (see Appendix 6.2). However, to observe the known *Best* solution, higher deviations are sometimes required. Typically, for HEC – during one run, a deviation of 50% returns the best-known solution of 557 instances of students sitting back-to-back exams. Otherwise, with this data set, only smaller deviation measures are required to discover the known best. Similar statements can be said for EAR and TRENT. Generally though, the percentage errors from the infinite *Best* are small and the time saving percentages for deviations below or equal to 10 are consistently large and using a dynamic deviation method is advised. To generalise, a deviation score of ten appears a good compromise between solution quality and computational effort.

6.2.10 Speed of Solution Return

This section extends the study of Section 6.2.9 and considers the computational effort required to first achieve salient or *Best* solutions while using the deviation concept discussed above.

It has been found that there is a relationship between the *Before TSP* and *After TSP* scores. Additionally, it has been proved that relatively good *After TSP* solutions can be discovered via low deviation scores. This, in consequence, encourages reduced runtimes due to the smaller number of feasible timetables that are improved through the TSP strategy. We have assumed the condition that the predetermined number of cycles will be completed. However, the scheduler may wish to alter this approach. The scheduler may allow for a diversification tactic such as: if the overall best solution has not been bettered for a number of iterations or a set time interval then trails are reinitialized and a new ant search is permitted. This section examines whether it is warranted to use higher deviations and allow for diversification in order to minimise computational expense. A subset of the percentage deviations has been selected (with *Infinite* representing the improvement of all feasible solutions, irrespective of *Before TSP* score) along with solutions equivocal to or better than the *Best* solution returned through 0% deviation. In order to make efficiency comparisons, the runtimes when each tabulated solution was observed are also presented.

The results are presented as follows.

For example, a best solution of 575 was found after 60.08 seconds

	0		1		5		10		20		Infinite	
Run	*Score*	*Secs*	*Score*	*Secs*	*Score*	*Secs*	*Score*	*Secs*	*Score*	*Secs*	*Score*	*Secs*
1	579	10.72	579	10.78	579	10.85	579	11.04	579	11.36	579	12.70
					574	38.27	574	48.89	574	78.65	574	142.11
2	584	20.74	584	20.93	584	21.76	584	24.13	575	20.06	575	26.46
					579	58.15	578	39.48	574	87.02	562	47.80
					578	91.85	576	91.75	571	458.65	557	71.80
							575	192.51				
							574	307.33				
							571	317.91				
3	575	60.08	561	22.63	561	24.68	568	19.13	568	23.93	550	32.49
					552	35.20	537	122.99	550	35.94	533	157.96
					550	89.47	533	132.45	533	222.56	512	528.36
					539	108.47	512	193.31	512	315.81		
4	561	31.30	526	24.74	526	25.76	526	26.86	520	24.32	520	45.60
5	617	31.47	617	32.04	617	36.91	580	10.07	580	10.33	580	11.79
					602	39.33						
					592	41.81						

Table 6.18 – Solution timing information for HEC.

	0		1		5		10		20		Infinite	
Run	*Score*	*Secs*	*Score*	*Secs*	*Score*	*Secs*	*Score*	*Secs*	*Score*	*Secs*	*Score*	*Secs*
1	973	86.14	973	86.25	973	86.70	973	87.37	973	89.29	973	91.20
			944	104.58	944	107.39	936	369.24	936	112.27	936	118.91
					936	277.52			935	214.58	919	130.55
2	1038	359.41	1038	360.42	1031	201.53	1031	234.62	1018	93.38	1018	96.65
							1023	249.47	1000	260.82	1000	341.07
							1019	294.69	990	604.36	990	980.50
3	1019	76.00	1019	76.11	1019	76.78	1019	78.47	1019	79.37	1019	79.37
			970	98.87	970	101.00	970	109.56	970	138.94	970	150.64
							948	626.43	948	935.38	948	976.34
4	991	274.46	991	275.93	981	111.49	981	114.42	981	122.07	981	129.16
					961	223.91	932	271.81	903	219.05	903	240.99
					958	349.52	905	417.86				
5	895	155.56	895	155.79	895	156.57	895	158.60	895	164.90	895	168.62
					868	204.27	861	222.17	852	644.21	852	689.12
							852	445.68	842	955.22	842	1024.65

Table 6.19 – Solution timing information for EAR.

	0		1		5		10		20		Infinite	
Run	*Score*	*Secs*	*Score*	*Secs*	*Score*	*Secs*	*Score*	*Secs*	*Score*	*Secs*	*Score*	*Secs*
1	1046	544.74	1042	435.19	1015	306.09	1015	341.55	998	546.64	973	586.04
			1028	453.72	1012	409.13	999	554.01	980	646.42	930	780.65
					999	457.35	997	581.47	930	703.52		
							989	614.01				
2	999	285.01	999	286.10	999	296.36	999	337.81	990	349.87	990	349.07
					969	373.45	996	403.06	969	587.06	969	587.06
							969	436.18	963	982.38	942	982.38
3	1016	556.02	1016	557.44	1015	348.03	1011	372.98	986	272.31	986	298.71
							998	758.26	965	627.39	965	714.23
4	986	460.47	986	462.98	957	274.92	957	299.24	953	368.51	986	199.24
											953	436.15
5	991	208.58	991	209.34	991	212.94	991	226.69	991	264.77	991	292.48
	990	513.05	990	515.12	990	539.78	983	622.85	980	276.19	980	306.95
									956	847.90	956	1048.53

Table 6.20 – Solution timing information for TRENT.

If the scheduler changes the focus of the search to allow diversification then the conclusions formed through Section 6.2.9 could alter. After analysing a breakdown of significant solutions and the clocked runtimes we notice that good, quickly produced solutions can be generated via small deviations. However, it is demonstrated that better solutions are often observed via larger deviations and still within a respectable time limit. This suggests that the production of one good *After TSP* timetable is discovered early in the search and its origins belong to a timetable of perceived lower *Before TSP* quality. This proves that some ant-constructed timetables (irrespective of the evidence in Figures 6.4-6.6) can be deemed of poor quality but reasoning can be attributed to a small of number of pairwise exams that are allocated to adjacent timeslots with a relatively high number of students in common.

It has been shown that using larger deviations leads to the observation of *Best* solutions at earlier stages of the search and the introduction of a stopping criterion/diversification tactic would lead to the minimisation of wasted computational effort. However, it is shown that a relationship exists between *Before TSP* and *After TSP* and consequently, the introduction of an inappropriate stopping criterion that prevents the construction of superior *Before TSP* timetables by stopping the search too early will potentially lead to poorer solution quality. Therefore, it is proposed here that no stopping criterion is imposed and lower deviations are used as proposed previously.

6.2.11 Baldwinian and Lamarckian systems

Until this juncture, the TSP strategy has been used to repair timetables with regards to second-order quality but subsequent ants have not used any feedback derived from this improvement phase (known as Darwinian theory). Within this section, we alter our focus slightly and allow the trail to incorporate information accumulated during the *Add-On* phase. Let us consider two approaches, Baldwinian and Lamarckian. The former allows the TSP to repair selected (dependent on deviation rule) ant-based solutions and, for a given timetable, the resultant new level of fitness is used within the feedback mechanism (see Equation 5.11). However, the pairwise exams that are updated originate from the *Before TSP* timetable. The latter also utilises the *After TSP* fitness score within the feedback mechanism, but the pairwise exams belonging to the *After TSP* timetables are updated. Table 6.21 presents *After TSP* second-order statistics relating to each of the evolutionary theories for the TSP.

Data Set	Strategy	Av	Best	Best 10	Worst 10	Feasible
HEC	*Darwin*	647.39	512	558.06	793.18	76.84
	Baldwin	684.51	526	621.20	789.04	4.16
	Lamarck	647.59	484	521.18	820.62	50.52
EAR	*Darwin*	1130.76	842	939.20	1413.02	62.83
	Baldwin	1198.21	941	1102.50	1293.76	2.46
	Lamarck	1250.36	892	969.76	1572.70	33.64
TRENT	*Darwin*	1135.61	930	971.38	1354.84	71.45
	Baldwin	1180.71	1009	1061.36	1316.14	33.18
	Lamarck	1179.67	919	996.90	1385.18	39.57

Table 6.21 – After TSP Statistics for different fitness strategies across all data sets.

For Baldwinian strategy, the volume of feasible timetables drops drastically. However, despite this low number, solution quality is maintained to a relatively acceptable level. An intuitive reason for this is as follows. Drastic solution changes are possible through the TSP and consequently, it is feasible to generate a significantly lower second-order score than originally achieved. However, the ant-based solution structure, along with the modified fitness, is used for trail feedback, which leads to mixed communication and consequently, greater exploration and potentially less exploitation. However, even though the feasibility rate is poor, good quality feasible timetables are recorded.

Lamarckian strategy encourages the production of relatively good timetables. The area of the solution space widens as a result of this evolutionary theory, which is indicated by the

indifferent average second-order scores but better *Best* solutions. The volume of feasible timetables falls, but not as severely as for Baldwinian. Given the first and second-order statistics, it is difficult to see any advantage in using Baldwinism or Lamarckism ahead of Darwinism in conjunction with TSP.

6.2.12 Data Sets with Side-Constraints

When an exam scheduler receives the data representing the exams to be timetabled it usually comes equipped with a list of side constraints requested by a specific lecturer or department. The most common demand is the allocation of an exam to a certain day or to a timeslot belonging to a subset of time periods. These are known as pre-assigned and time windowed exams respectively. The acceptance of such requests limits the effectiveness of the TSP strategy to reduce back-to-back conflicts. Since each timeslot/city contains a set of exams, if at least one exam is pre-assigned or time-windowed then the movement of that timeslot/city is severely restricted or non-existent. A number of such 'problematic' exams has the potential to reduce the effectiveness of the TSP model.

To illustrate this discussion, two real-life data sets are used and comparisons between the unrestricted and restricted cases are made. Data representing January 2000 and summer 2002 exams, Swan2000 and Swan2002 respectively, at the University of Wales Swansea have the following characteristics:

	Swan2000	Swan2002
Exams	313	722
Timeslots	20	36
Students	4611	6388
Density	10.33%	4.20%

Table 6.22 – Data set characteristics for Swan2000 and Swan2002.

The parameters were levied as follows:

	Parameter	Setting
Within Algorithm	Ants	313/722
	Cycles	100
	α	2
	β	1
	ρ	0.5
	δ	100
	δ_2	650
	σ	30
	e	15
Improvement - TSP	Ants	30
	Cycles	30
	α_{tsp}	1
	β_{tsp}	1
	ρ_{tsp}	0.5
	Q_{tsp}	100
	e_{tsp}	3
	σ_{tsp}	15

Table 6.23 – Parameter information

The algorithm was run, with the above parameter settings, and two scenarios were considered – the restricted and the unrestricted case.

Illustration of the restricted case

The reorganisation of the timeslots is restricted by pre-assigned and time windowed exams. Tables 6.24 and 6.25 detail the number of exams pre-assigned to each timeslot for the Swan2000 and Swan2002 data sets respectively.

Timeslot	*Exams*	*Timeslot*	*Exams*	*Timeslot*	*Exams*	*Timeslot*	*Exams*
1	0	6	0	11	1	16	0
2	1	7	0	12	0	17	1
3	1	8	0	13	2	18	1
4	1	9	1	14	1	19	0
5	0	10	1	15	1	20	2

Table 6.24– Information regarding the frequency of preassigned exams per timeslot for Swan2000.

With respect to Swan2000, 12 timeslots are fixed in each timetable due to the demands associated with 14 exams. The scheduler has the flexibility of reallocating only 8 'free' timeslots. It is possible to move timeslots 1, 5, 6, 7, 8, 12, 16 and 19 without violating these pre-assignment requests. Additionally, there are 20 time windowed exams that can, on average, be placed in 8 timeslots.

Timeslot	*Exams*	*Timeslot*	*Exams*	*Timeslot*	*Exams*	*Timeslot*	*Exams*
1	3	10	1	19	3	28	1
2	0	11	3	20	1	29	3
3	1	12	2	21	6	30	3
4	0	13	3	22	2	31	0
5	2	14	3	23	2	32	0
6	2	15	1	24	4	33	3
7	2	16	0	25	2	34	0
8	0	17	4	26	2	35	0
9	3	18	1	27	4	36	0

Table 6.25– Information regarding the frequency of pre-assigned exams per timeslot for Swan2002.

With respect to Swan2002, 27 timeslots are fixed in each timetable. The scheduler is allowed to shift only 8 timeslots within the timeslot sequence due to the demands of 67 pre-assigned exams. Despite the increased number of timeslots, less flexibility exists in comparison to Swan2000. Also, there are 47 time-windowed exams that can, on average, be placed in approximately 26 timeslots.

Reduced flexibility does reduce the size of the problem and consequently, limits the number of cities that need to be considered. The lower the number of cities, the bigger the argument for an exact based TSP technique. However, it has been stated in Section 6.2.1 that since the TSP is the traditional testbed for ACO algorithms it is warranted here to treat the TSP as a second combinatorial optimization problem. Therefore, it is justified to apply ants in all such cases given the nature of this study.

Results

Table 6.26 presents the results for both scenarios for Swan2000 and Swan2002. The reduction in flexibility results in higher second-order conflict scores as expected.

		Av	**Best**	**Best 10**
Swan2000	*Restricted*	341.45	121	124.61
	Unrestricted	257.92	65	86.65
Swan2002	*Restricted*	705.88	331	375.81
	Unrestricted	661.40	193	235.50

Table 6.26– Statistics comparing solution quality for restricted and unrestricted case.

The inclusion of side constraints does inhibit the success of the TSP but not as severely as the evidence in Tables 6.24 and 6.25 suggest.

It has been shown that the TSP is capable as an improvement strategy. Large enhancements are possible and correspond to the positive conclusions drawn by authors such as White and Chan (1979) and Colijn and Layfield (1995). However, it has been demonstrated that a scheduler can be disadvantaged when side constraints, which limit the flexibility of the TSP, exist.

6.3 Exam exchange methods

The previous section described how the order of the timeslots could be re-arranged via the TSP to reduce the second-order conflicts score of the timetable. It regarded each timeslot as a single entity and no within entity alterations were permitted. Although, the quality of the timetables did improve due to TSP involvement, the method itself is a little restrictive. Exam movements, rather than timeslot movements, is intuitively more flexible. A greater permutation of timetables from one starting solution (ant constructed solution) is viable leading to a greater probability of achieving superior final solutions. In this section, we will discuss two exam exchange strategies. Firstly, a relatively simple local search variant is used. Exams that inflate the second-order score the greatest are recognized and placed (potentially) within a feasible timeslot that will consequently reduce the overall second-order score of the timetable by the greatest amount. Secondly, exam movements based on the graph theoretic Kempe Chains is discussed. As in Section 6.2, each technique is subjected to a series of studies. For both techniques, the influence of the three evolutionary theories is assessed and means of reducing the usage of these exam movement *Add-Ons* is investigated.

6.3.1 Local search

One form of local search is Hill-Climbing. This method will not accept worsening moves and only has the ability to search a very limited section of the solution space. It is however fast compared to other improvement methods and will not produce a solution worse than the original. For these experiments, both Random Descent and Steepest Descent were considered and it was observed that the latter encouraged the better results with the former requiring less runtime. Given that runtimes for both descent methods are significantly lower than experienced with methods presented in Sections 6.2 and 6.3.2, it was felt wise to proceed with the strategy that generated superior solution quality – Steepest Descent.

Darwinian, Baldwinian and Lamarckian evolutionary theories are applied and compared with respect to this descent method. In Section 6.2, we used terms such as *Before TSP* and *After TSP*, these will be replaced with *Before Loc* and *After Loc*.

6.3.1.1 Steepest Descent

Steepest Descent is defined as follows. A neighbourhood of a solution x is denoted by $N(x)$ and members of $N(x)$ are those solutions that can be reached by making one sequential change to x while retaining solution feasibility. Every member of $N(x)$ is examined and the exam move that creates the biggest reduction in second-order conflict scores is selected and a new solution x is defined. The procedure continues until no improving moves are possible. Local search is defined in greater depth in Appendix 6.3.

After Loc statistics for the three evolutionary theories are presented in Table 6.27.

Data Set	Strategy	Av	Best	Best 10	Worst 10	Feasible	Std Dev
HEC	*Darwin*	726.18	587	654.80	1359.26	76.84	66.57
	Baldwin	646.94	552	623.62	1502.85	77.76	95.53
	Lamarck	679.58	575	646.95	1523.61	75.22	91.59
EAR	*Darwin*	1053.22	826	947.92	1436.36	62.83	65.22
	Baldwin	946.73	819	922.12	1554.69	67.91	88.08
	Lamarck	941.49	799	892.62	1554.43	66.58	89.27
TRENT	*Darwin*	960.40	770	831.98	1426.06	71.45	83.63
	Baldwin	889.60	750	827.78	1540.15	72.68	94.79
	Lamarck	883.63	740	820.59	1522.45	71.99	97.91

Table 6.27– Statistics describing After Loc performance of Steepest Descent strategy for three evolutionary theories.

An improvement in solution quality is observed when repair information is integrated, to some degree, in the trail. *Av*, *Best* and *Best 10* statistics provide indication that Baldwinian and Lamarckian fitness philosophies enhance the second-order fitness of the timetables and future ant based searches should consider these strategies. The first-order performance of the searches is fairly consistent across the evolutionary theories for all data sets. The standard deviation of the second-order conflicts score has been recorded. This statistic has been averaged over 5 independent runs. Generally, the standard deviations for Baldwin and Lamarck are higher than for Darwin. Therefore, it is perceivable that a greater range of second-order scores is generated through Baldwinian and Lamarckian strategies (further illustrated by the difference in *Best10* and *Worst 10* statistics). This suggests that Baldwinian and Lamarckian philosophies broaden the investigation around the local

minima and consequently, increase the chances of forming a timetable of relatively superior second-order quality.

Since Baldwinian and Lamarckian philosophies impose alternative fitness strategies, the constructions drafted by the ants (*Before Loc* statistics) will differ. Consequently, we will analyse the *Before Loc* statistics for each of the fitness techniques.

Data Set	Strategy	Av	Best	Best 10	Worst 10	Std Dev
HEC	*Darwin*	935.66	673	711.28	2237.94	195.71
	Baldwin	1048.42	611	644.52	2427.58	307.66
	Lamarck	1102.76	610	663.62	2403.10	288.12
EAR	*Darwin*	1296.94	1037	1104.02	1788.46	104.66
	Baldwin	1225.66	950	1017.38	1784.80	109.48
	Lamarck	1241.08	935	999.00	1807.14	112.29
TRENT	*Darwin*	1259.89	985	1039.48	1801.28	120.51
	Baldwin	1204.11	894	963.84	1792.60	124.53
	Lamarck	1189.43	904	960.04	1767.02	126.36

Table 6.28– Statistics describing Before Loc performance of Steepest Descent strategy for three evolutionary theories.

The results are promising. The use of Baldwinian and Lamarckian theories do generate superior solutions - we note the improved *Best* and *Best10* results. However, there is no discernable improvement in *Av* scores and a deterioration in the *Worst10* statistics, which suggest that the ant searches widen. This intuition is consolidated by the standard deviation of second-order scores (for feasible timetables) statistics. These figures suggest that Baldwin and Lamarck not only avoid searching in a smaller subset of the solution space but actually encourages the ants to widen the search (Section 6.3.2.5 performs a study regarding the width of exploration) around local minima. This leads to a greater variety of timetables produced and consequently, a wider variety of *After Loc* timetables, which further explains the results presented in Table 6.28.

6.3.1.2 Relationship between *Before Loc* and *After Loc*

Here we revisit the style of investigation performed in Section 6.2.8 – to determine if a relationship between *Before Loc* and *After Loc* exists and consequently, (if appropriate) exploit this relationship as to reduce the number of timetables that are repaired, thus lowering computational effort. For each data set, the *Before Loc* and the corresponding *After Loc* scores across all runs were collated and ordered according to *Before Loc*. Both variables were grouped according to *Before Loc* into intervals of size 100 and averaged.

Spearmans Rank Correlation statistic is 0.24, 1 and 1 for HEC, EAR and TRENT respectively. The corresponding p-values of significance are 0.02, 0 and 0 respectively. Thus, each of the Spearmans statistics is deemed as statistically significant despite the relatively low correlation statistic for HEC i.e. 0.24.

Representing *Before Loc* and *After Loc* graphically (not shown here to avoid repetition of section 6.2.8) shows that the relationship appears closer to linear than any other form. Consequently, to quantify the effect of *Before Loc* on *After Loc*, least squares regression has been performed and can be viewed in Appendix 6.4. Given the positive trend between the variables, it seems logical to apply local search to the fitter timetables in order to reduce the computational effort required. Section 6.3.1.3 will investigate this. For completeness, the relationship between *Before Loc* and the reduction in second-order conflicts (*Saving*) is analysed. Like Section 6.2.8, the relationship is strongly linear (can be shown graphically) across the data sets. Computation of Spearmans Rank produces non-parametric correlations of 1 for each of the data sets. This highlights that the Steepest Descent Strategy can repair poorer solutions substantially, however, to produce solutions of 'better' *After Loc* quality requires 'better' *Before Loc* solutions.

6.3.1.3 Quality-Time Trade-Off

Section 6.3.1.2 demonstrated that a strong positive relationship between *Before Loc* and *After Loc* is present. In consequence, this suggests that only a subset of fitter *Before Loc* timetables are needed to produce a 'successful' timetable. The study discussed in Section 6.2.9 will be applied here. After the construction of each feasible timetable, the second-order conflicts score is computed. If it lies within a percentage deviation of the run minimum for feasible timetables then local search is used to repair the timetable. Each of the fitness philosophies are analysed since reducing the number of repairs will bear an influence on the trail levels when used with Baldwin and in particular, Lamarckian strategies and consequently, have an impact on solution quality. The results for each of the data sets are tabulated in Tables 6.29-6.31.

Run	% Dev Run Min	Av	Best	Best 10	Worst 10	Eligible	Time	% Dev Best	% Dev Time
DARWIN	Infinite	726.18	587	654.80	1359.26	7684	166.65	-	-
	0	763.24	658	707.78	812.98	18.40	84.05	12.10%	-49.57%
	1	708.46	657	685.16	862.36	223.00	86.25	11.93%	-48.24%
	3	702.33	645	679.12	868.12	634.60	90.69	9.88%	-45.58%
	5	703.09	645	677.22	877.56	1228.00	97.08	9.88%	-41.75%
	10	706.31	638	665.64	900.74	2698.20	112.93	8.69%	-32.24%
	20	710.21	631	661.64	965.68	4181.20	128.91	7.50%	-22.65%
	30	714.31	631	660.86	1025.24	5176.00	139.63	7.50%	-16.22%
	50	720.19	631	659.54	1085.68	6195.80	150.62	7.50%	-9.62%
BALDWIN	Infinite	646.94	552	623.62	1502.85	7776.80	189.33	-	-
	0	826.23	612	786.78	961.27	18.60	102.62	10.87%	-45.80%
	1	730.70	648	753.96	951.07	104.60	104.65	17.39%	-44.73%
	3	713.82	606	733.75	970.13	218.20	104.03	9.78%	-45.05%
	5	689.35	596	729.64	998.59	636.00	108.29	7.97%	-42.80%
	10	701.18	558	724.43	1024.16	887.00	110.26	1.09%	-41.76%
	20	655.50	584	695.28	1064.62	3280.40	136.95	5.80%	-27.67%
	30	674.47	588	701.96	1167.60	4125.00	148.84	6.52%	-21.39%
	50	**643.97**	**551**	**651.55**	**1285.17**	**4471.00**	**154.35**	**-0.18%**	**-18.48%**
LAMARCK	Infinite	679.58	575	646.95	1523.61	7521.80	192.04	-	-
	0	826.23	612	786.78	961.27	18.60	102.75	6.43%	-46.50%
	1	725.96	582	747.73	972.15	59.20	104.66	1.22%	-45.50%
	3	712.97	622	739.79	990.22	50.00	108.15	8.17%	-43.68%
	5	709.82	618	729.83	1022.99	375.40	107.96	7.48%	-43.78%
	10	669.69	594	706.82	1043.45	1848.70	124.93	3.30%	-34.95%
	20	665.65	587	691.18	1093.82	2355.20	128.35	2.09%	-33.16%
	30	682.75	586	707.87	1154.14	2296.60	131.90	1.91%	-31.32%
	50	692.42	573	702.26	1235.72	4849.80	158.87	-0.35%	-17.27%

Table 6.29 – Statistics describing achievable results for dynamic min+deviation rule for HEC.

Run	*% Dev Run Min*	*Av*	*Best*	*Best 10*	*Worst 10*	*Eligible*	*Time*	*% Dev Best*	*% Dev Time*
DARWIN	Infinite	1053.22	826	947.92	1436.36	6283.00	718.86	-	-
	0	1053.47	826	1005.06	1096.00	17.40	374.76	0%	-47.87%
	1	1048.99	826	982.08	1149.84	41.00	376.05	0%	-47.69%
	3	1029.62	826	967.24	1185.72	162.20	382.71	0%	-46.76%
	5	1025.00	826	961.64	1215.68	436.00	397.75	0%	-44.67%
	10	1029.62	826	954.98	1250.96	1889.40	477.57	0%	-33.57%
	20	1042.39	826	949.40	1296.30	4950.20	645.66	0%	-10.18%
	30	1049.80	826	948.80	1309.16	5929.40	699.44	0%	-2.70%
	50	1052.97	826	947.92	1310.80	6275.20	718.43	0%	-0.06%
BALDWIN	Infinite	946.73	819	922.12	1554.69	6791.00	678.35	-	-
	0	1156.95	907	1154.03	1257.62	16.40	305.37	10.74%	-54.98%
	1	1114.99	871	1111.31	1276.12	25.60	306.75	6.35%	-54.78%
	3	1040.25	855	1058.31	1323.10	119.00	309.65	4.40%	-54.35%
	5	1001.84	863	1055.24	1341.18	420.20	328.94	5.37%	-51.51%
	10	937.71	770	976.54	1398.07	2074.60	415.21	-5.98%	-38.79%
	20	971.21	784	987.75	1474.27	4522.80	549.27	-4.27%	-19.03%
	30	**954.76**	**757**	**970.53**	**1523.08**	**5853.40**	**623.93**	**-7.57%**	**-8.02%**
	50	944.45	801	954.70	1574.05	6774.60	676.80	-2.20%	-0.23%
LAMARCK	Infinite	941.49	799	892.62	1554.43	6658.20	732.53	-	-
	0	1124.31	834	1107.10	1255.57	17.80	325.82	4.38%	-55.52%
	1	1074.35	828	1080.91	1256.85	33.00	326.49	3.63%	-55.43%
	3	997.98	836	1039.18	1244.01	100.00	332.98	4.63%	-54.54%
	5	983.12	834	1021.28	1298.38	255.80	343.49	4.38%	-53.11%
	10	951.21	829	991.19	1362.82	1546.20	412.50	3.75%	-43.69%
	20	937.79	781	960.86	1466.85	4318.00	582.28	-2.25%	-20.51%
	30	928.04	793	933.80	1535.64	6061.20	698.48	-0.75%	-4.65%
	50	945.58	833	954.76	1582.40	6677.20	727.94	4.26%	-0.63%

Table 6.30 – Statistics describing achievable results for dynamic min+deviation rule for EAR.

Run	% Dev Run Min	Av	Best	Best 10	Worst 10	Eligible	Time	% Dev Best	% Dev Time
DARWIN	Infinite	960.40	770	831.98	1426.06	7145.00	2096.15	-	-
	0	971.72	814	900.62	1052.48	22.40	633.39	5.71%	-69.78%
	1	962.66	814	876.28	1106.84	47.40	638.52	5.71%	-69.54%
	3	933.65	808	855.80	1143.72	164.60	662.59	4.94%	-68.39%
	5	923.67	800	848.42	1159.16	416.00	714.22	3.90%	-65.93%
	10	922.21	775	836.96	1208.92	1851.20	1008.97	0.65%	-51.87%
	20	940.72	770	832.32	1288.04	5093.60	1674.86	0.00%	-20.10%
	30	953.43	770	832.18	1375.60	6620.80	1988.50	0.00%	-5.14%
	50	959.98	770	831.98	1329.76	7131.40	2093.36	0.00%	-0.13%
BALDWIN	Infinite	889.60	750	827.78	1540.15	7268.80	2257.82	-	-
	0	1066.49	843	1039.33	1214.93	19.40	778.93	12.40%	-65.50%
	1	1013.89	837	997.06	1230.94	37.40	782.49	11.60%	-65.34%
	3	930.66	751	927.19	1258.19	124.60	800.43	0.13%	-64.55%
	5	919.69	774	928.83	1299.78	250.20	822.69	3.20%	-63.56%
	10	859.94	735	868.67	1387.00	1137.80	997.56	-2.00%	-55.82%
	20	**863.85**	**712**	**851.55**	**1465.91**	**4454.80**	**1663.53**	**-5.07%**	**-26.32%**
	30	867.58	728	853.03	1538.59	6132.80	1995.03	-2.93%	-11.64%
	50	884.78	744	858.51	1522.24	7238.40	2252.43	-0.80%	-0.24%
LAMARCK	Infinite	883.63	740	820.90	1522.45	7199.20	1526.95	-	-
	0	1051.76	794	1011.70	1203.30	18.40	536.04	7.30%	-64.89%
	1	1029.16	829	993.57	1251.34	37.60	538.76	12.03%	-64.72%
	3	942.12	783	939.57	1269.00	123.40	553.04	5.81%	-63.78%
	5	909.83	769	898.75	1317.99	274.60	574.21	3.92%	-62.39%
	10	867.52	756	866.59	1354.34	1146.00	682.95	2.16%	-55.27%
	20	880.32	761	877.65	1472.31	3771.60	1020.6	2.84%	-33.16%
	30	883.42	763	858.70	1509.65	6011.00	1340.01	3.11%	-12.24%
	50	880.29	747	850.47	1555.83	7061.60	1483.31	0.95%	-2.86%

Table 6.31 – Statistics describing achievable results for dynamic min+deviation rule for TRENT.

It is shown that Baldwinian evolution is the superior evolutionary theory. This can be attributed to the restrictive nature of Steepest Descent and is explained in greater depth in Section 6.6. Additionally, Lamarckian theory outstrips Darwinian. Limiting improvement (to selected ant-based solutions) does, on occasions, generate superior solutions than improving every feasible solution (particularly with Baldwinian theory). This suggests that the use of repair information in more limited measures can actually lead to better *Best After Loc* solutions. This indicates that the improvement of a smaller number of good ant-based solutions allows for a greater differentiation between solutions, thus enabling the discovery of better solutions. Similar statements can be made regarding *Before Loc* solutions (See Appendix 6.5). For HEC, at 50% deviation with Baldwinian theory, a *Before Loc* observation of 580 is noted, but with infinite deviation, the best observation is 611 students. For EAR, at 10% with Baldwinian theory, a best of 875 is noted but the best infinite deviation result is 950. Similar statements can be made regarding Lamarckian.

6.3.1.4 Comparison of Local Search with TSP

Here we compare the TSP and Local Search methods with regards to solution quality. TSP produces better solutions for HEC, but does not compete when dealing with EAR and TRENT. Intuitively, it can be explained as follows. HEC is the denser graph and the feasible reallocation of individual exams is more difficult due to the greater number of direct conflicts. Meanwhile, the rearrangement of timeslots does not, at any time, violate these first-order concerns. Consequently, such freedom associated with the TSP leads to better results. However, research has shown that solution quality with the less dense EAR and TRENT data sets is superior with the exam exchange approach.

Additionally, the computational effort required for local search is less. This is to be expected due to the nature of these improvement strategies. Steepest Descent local search terminates when the local minimum is obtained. The ANT-TSP system, even though quicker than exact TSP, has a predetermined number of cycles to complete. Stopping the algorithm after n non-improving cycles would reduce runtimes but local search would still be more efficient. It should be noted here that much computational expense could be saved if Steepest Descent was used to re-order the timeslots. However, the desire behind this thesis is to use ants as much as possible and provides reason why the ANT-TSP algorithm has been applied. Local search and, in the next main section, Kempe chain descents have been used to move individual exams since it is not feasible to use ants for such a task.

6.3.1.5 Further Illustration of Role of Ants

It has already been demonstrated that the quality of the ant constructed solution influences the quality of the final solution (after an enhancement technique has been applied). Consequently, this highlights the critical role that the ants play. To further appreciate the importance of the ants, extra experiments have been performed.

The algorithm was run allowing solely first-order fitness as the trail feedback mechanism (see Chapter 4). As before, following the construction of a feasible timetable, local improvement is performed. The Lamarckian Steepest Descent technique was applied and the second-order results were collated. These are presented below.

Data Set	Av	Best	Best 10	Worst 10
HEC	913.05 (20.47)	633 (7.27)	710.94	1539.36
EAR	1331.77 (20.92)	951 (13.14)	1117.58	1749.12
TRENT	1004.19 (4.36)	813 (5.29)	856.20	1591.76

Table 6.32 – Statistics describing solution quality with Steepest Descent applied to poorer starting solutions.

The terms in parenthesises refer to the percentage change in second-order quality between these experiments and the tests that incorporated the second-order fitness feedback (Table 6.28, Darwin rows). Table 6.32 indicates the importance of better ant-based solutions to the overall performance of the search. A more efficient ant-based search generates lower *After Loc* second-order conflict scores. With HEC, the average solution score improves by a factor of 20.47%, with EAR, 20.92% and with TRENT, 4.36%. Meanwhile, the best solutions improve by 7.27%, 13.14% and 5.29% for HEC, EAR and TRENT respectively. These improvements illustrate the importance of encouraging the ants to construct good second-order feasible timetables, highlighting the importance of the efficiency of a good ant based search, which validates the role of the ants.

6.3.1.6 TSP – Local Search Hybrid (TSP-Local)

In this section, we consider a TSP and Steepest Descent hybrid, see for example White and Chan (1979) and Colijn and Layfield (1995). Such a collaboration of techniques will allow the movement of both individual exams and timeslots. This method allows room for further improvement since it eliminates some of the restrictions of the methods when applied independently. Typically, the TSP approach has a main drawback of not permitting the reallocation of exams to alternative timeslots, while the Steepest Descent strategy terminates when no improving individual movements can be performed.

After the construction of a feasible timetable, the timeslots are reorganised via the TSP strategy. After the predetermined number of TSP cycles has been fulfilled, improvements, if possible, are made through applying Steepest Descent to the movement of individual exams. The order of strategy application has been chosen to minimise runtimes. Since the TSP technique completes a predetermined number of cycles and the Steepest Descent strategy terminates as soon as no further improvements are possible, it seems realistic to experience shorter runtimes through this approach.

Darwinian, Baldwinian and Lamarckian philosophies are again applied. The results are tabulated below:

Data Set	Strategy	Av	Best	Best 10	Worst 10	Feasible
HEC	*Darwin*	588.50 (18.96)	467 (20.44)	506.18	719.16	77.40
	Baldwin	567.12 (12.34)	446 (19.20)	483.86	724.94	70.80
	Lamarck	573.38 (15.63)	427 (25.74)	461.74	729.04	51.35
EAR	*Darwin*	921.64 (12.49)	722 (12.59)	772.28	1143.30	63.24
	Baldwin	892.50 (5.73)	724 (11.60)	756.92	1126.60	63.47
	Lamarck	937.54 (0.42)	720 (9.89)	759.72	1211.84	29.24
TRENT	*Darwin*	886.34 (7.71)	724 (5.97)	766.96	1059.92	70.71
	Baldwin	870.55 (2.14)	715 (4.67)	742.00	1058.82	72.98
	Lamarck	901.56 (2.03)	725 (2.03)	768.58	1059.74	26.82

Table 6.33 – Statistics describing final solution quality for the TSP-LOCAL hybrid strategy.

The bracketed terms represent the percentage change in solution quality improvement when comparing the TSP-Local hybrid to the Steepest Descent strategy (Table 6.27). Larger improvements are experienced for HEC and EAR, while limited quality enhancement is experienced with TRENT. Generally, the hybrid strategy assists Darwinian philosophy greater than the other evolutionary strategies. This can be attributed to the nature of these theories. It has been demonstrated that the ants could construct better *Before Add-On* (*Before Add-On* and *After Add-On* are generalized terms for the second-order scores recorded before and after improvement is applied to a timetable) solutions if *After Add-On* feedback (Baldwin and Lamarck) is passed to the trail. Therefore, this leads to better *After Add-On* solutions due to our conclusions regarding the relationship between *Before Add-On* and *After Add-On* solutions. Thus, the benefit that can be derived from TSP based improvement (in addition to local search) is going to be smaller for Baldwinian and Lamarckian systems than Darwinian. Discussion of runtimes can be found later in this chapter.

The use of this hybrid is warranted, as the following statistics will testify. With HEC, a best of 427 (Lamarck) with TSP-Local compares with a best of 551 (Baldwin) with Local

and 484 (Lamarck) with TSP. With EAR, a best of 720 (Lamarck) with TSP-Local contrasts with a best of 757 (Baldwin) with Local and 842 (Darwin) with TSP. With TRENT, a best of 715 (Lamarck) measures against a best of 712 (Baldwin) with Local and 919 (Lamarck) with TSP. Despite a marginally better performance (for TRENT) with Local over TSP-Local, it is worth noting the *Best 10* statistics of the relevant methods (TSP-Local and Local), which are 742.00 and 851.55 respectively. These statistics indicate the benefit of the TSP-Local hybrid. On a generic level, Lamarckian theory is the superior method when regarding the TSP-Local hybrid. One reason can be attributed to the wider exploration that derives from Lamarckian evolution (as demonstrated later in this chapter).

6.3.2 Kempe Chains

6.3.2.1 Introduction

An *Add-On* uses the ant-based solution as a starting solution and searches the neighbourhood. Since neighbourhood structure will have a direct influence upon the quality of the final solution, its definition plays a critical role. In this chapter, we have already detailed local search but, the solution space landscape is spiky and the ability to reach some solutions is limited. Thompson (1995) discussed means to extend the neighbourhood structure and enhance the reachibility of some solutions. Thompson claimed that the use of a graph theoretic exam exchange method, namely Kempe Chains, improved solution quality. Kempe Chains has the same influence as flattening the terrain around solutions, thus allowing for greater exploration. Also, Morgenstern and Shapiro (1989b) stated that neighbourhood definition is more important than the nature of the solution method and therefore, a suitable *Add-On* technique here should have benefits. Johnson et al. (1991) stated that the implementation of Kempe Chains does require additional computational effort but the trade-off with solution quality compensates for this.

An ideal neighbourhood should ensure that solutions are reachable, yet be small enough to be searched efficiently. Morgenstern (1989a) discussed a method that swapped a number of vertices in two colours while maintaining feasibility. If a vertex v_r is to be moved from colour class c_1 to c_2, the move is only feasible if no vertices adjacent to v_r are currently coloured c_2. Exchange moves would be feasible if only one such vertex v_s in c_2 existed and

provided no other vertices in c_1 were adjacent to v_s. Kempe Chains were introduced by Kempe (1879) and can be defined as any connected component of the subgraph consisting of only the vertices in two distinct colour classes. To form a Kempe Chain, the subgraph consisting of only the vertices in two given classes is produced. Exchanging the colours of all vertices from a feasible colouring produces another feasible colouring. Consequently, a neighbourhood based on Kempe Chains will maintain feasibility and acts as a suitable improvement strategy for feasible ant constructed solutions.

A move consists of swapping the colours of the vertices in the Kempe Chain such that those coloured c_1 are now coloured c_2 and those coloured c_2 are now coloured c_1. As stated above, all exchanges will not violate feasibility. Let us consider a member v_r of a Kempe chain. If a vertex in v_r's new class is adjacent to v_r then it is deemed infeasible. However, all vertices in v_r's new colour class and neighbours of v_r are recoloured, which means the new colouring will be feasible.

Each Kempe Chain is sampled as in Johnson et al. (1991). Two colours c_1 and c_2 are randomly chosen as well as a single vertex v_r from c_1. Adjacent vertices to v_r coloured c_2 are noted and added to the chain. The next vertex on the chain is examined and neighbours in c_1 and c_2 are added to the chain. This procedure continues until the chain is complete and no members of the chain have a neighbour in c_1 or c_2. The Kempe Chain is labeled (c_1, c_2, v_r) which represents the chain with colour classes c_1 and c_2 and beginning with vertex v_r. Thompson (1995) showed that the above sampling procedure was the more appropriate and will be used here.

A solution achieved after application of Kempe chains is referred to as *After Kempe*, while the ant-based solution is labelled *Before Kempe*.

6.3.2.2 S-Chains

Morgenstern (1989a) indicated that the neighbourhood structure could be extended further through the use of S-Chains. Whereas Kempe chains are connected components of the subgraph induced by two colour classes, S-Chains originate from S different colour classes, where S≥2.

S distinct colours are randomly generated, with $2 \leq S \leq q$ (with q equal to the number of colours). A starting vertex is chosen at random from the first colour group and all adjacent vertices to the current vertex and in the next colour group are identified. This continues with the next vertex and the chain is completed when no further additions can be made to the S^{th} colour class.

The neighbourhood structure of S-Chains is defined as follows. A solution is achieved if every member of the S-Chain is moved forward one colour class. A neighbouring solution will always be feasible as any adjacent vertex in the subsequent colour class will itself be recoloured, thus maintaining feasibility.

Increasing S lowers the manageability of the neighbourhood and encourages extortionate computational times. Additionally, S-Chains can make large solution changes but can miss small, very good solution changes. Conclusions regarding S-Chains in Thompson (1995) suggest that the simpler Kempe Chains are better on both computational and solution quality fronts and therefore, Kempe Chains rather than S-Chains will be used here.

6.3.2.3 Application to Examination Timetabling

Moves in the Kempe Chain neighbourhood retain feasibility. This observation fits nicely within the examination timetabling framework. Since Kempe Chains are applied to feasible ant based solutions, the resultant timetables will be first-order feasible. Additionally, the large changes that are possible to solutions could result in large drops in second-order conflict scores. However, it is impossible to deduce in advance whether the generated timetables will be feasible with respect to binding constraints such as seating capacities. Consequently, much computational effort may be wasted generating infeasible moves.

If a Kempe chain contains a time-windowed exam vertex then a timeslot vertex will be added to the chain. Furthermore, another timeslot vertex will be added as the timeslot vertices form a clique. A Kempe chain move will exchange the colours of the timeslot vertices. The colouring will still be feasible because the time-windowed exam vertex will still be the same colour as the timeslot vertex.

6.3.2.4 Sample Size

The starting vertices of Kempe chains are randomly sampled. A sampled Kempe chain is deemed successful if the second-order score of the timetable is better than previously experienced. If no improved solutions are observed with a sample of size n then no more chains are sampled until the next applicable timetable (feasible). Conversely, if improvement is observed, exam swaps between timeslots are performed and chains relating to the best-improved solution are sampled. In this section, we observe the influence of sample size n with respect to both solution quality and computational effort. The statistics refer to Kempe modified timetables and are presented in Table 6.34.

	Sample n	Av	Best	Best 10	Worst 10	Time
HEC	*10*	874.05	624	749.71	1950.92	106.53
	50	733.04	508	642.44	1109.93	370.09
	100	686.40	501	609.13	977.26	786.90
	200	640.04	477	574.24	888.04	1635.85
	500	608.35	466	547.26	824.56	3635.09
	1000	592.85	438	535.91	804.32	6305.13
EAR	*10*	1213.72	932	1158.49	1818.94	292.29
	50	1110.72	879	1044.55	1534.51	1050.43
	100	1030.19	823	979.97	1407.18	2184.36
	200	967.89	780	914.71	1304.78	4882.73
	500	921.70	722	869.42	1231.07	13484.70
	1000	886.19	708	841.16	1171.92	25494.20
TRENT	*10*	1241.64	978	1164.85	1808.58	473.07
	50	1119.92	927	1076.44	1504.35	1685.47
	100	1050.06	839	994.44	1409.89	4958.45
	200	978.72	801	931.61	1304.21	11581.40
	500	916.15	751	878.56	1208.74	30640.50
	1000	891.96	736	861.05	1150.96	56021.30

Table 6.34 – Statistics describing the influence of Kempe sample size on solution quality and runtime.

The trade-off between solution quality and computational effort is clear. Large sample sizes require relatively expensive runtimes and the suitability depends on the time a scheduler is prepared to allow. Second-order enhancement is certainly noted through sampling more chains but relatively greater gains are observed for increases in n when n is small. Due to the runtime excess for high n, it has been decided to set $n=100$ for the remainder of this study.

TSP and Local Search procedures have shown comprehensively that there exists a relationship between the second-order scores of ant-based and *Add-On* solutions. This emphasises the importance of the quality of the ant solution and, as investigated

previously, the application of the computational expensive Kempe Chains should be limited to the better ant-based solutions.

In addition, there is a strong positive relationship between quality improvement and computational effort. Appendix 6.6 presents scatter diagrams that represent this relationship for each of the data sets. The Spearmans Rank correlation statistics are 0.67, 0.98 and 0.96 for HEC, EAR and TRENT respectively, which emphasises the strength of association. In consequence, if time limits are imposed on an investigation, this relationship suggests the pertinence of applying an *Add-On* to fitter ant based solutions.

6.3.2.5 Restricting Kempe Improvements

In this section, we consider the influence of limiting the number of timetables that are improved through Kempe Chains. As in Sections 6.2 and 6.3, the run-minimum is observed and Kempe Chains are applied to those feasible timetables that have second-order scores within a percentage deviation of the run-minimum. Additionally, the three evolutionary theories, Darwinism, Baldwinism and Lamarkism, are scrutinized. Only one entry (for infinite deviation) per data set is regarded for Darwin to reduce the amount of data presented. With respect to Darwinism, the best *Best* statistic is always recorded when infinite deviation is applied. Consequently, no better solutions are missed by not recording the deviation related statistics. However, a time versus solution quality trade-off becomes impossible. Since it has been noted that Baldwinism and Lamarckism are the preferred evolutionary theories (by previous investigation), the benefit of a detailed breakdown of Darwin-deviation statistics becomes obsolete. The results are as follows:

Run	% Dev Run Min	Av	Best	Best 10	Worst 10	Eligible	Time	% Dev Best	% Dev Time
DARWIN	Infinite	686.40	501	609.13	977.26	7684.00	786.90	-	-
BALDWIN	Infinite	683.60	494	567.60	971.50	7117.80	871.10	-	-
	0	713.31	572	710.73	784.89	17.60	68.14	15.79	92.18
	1	660.16	572	667.30	784.62	83.60	72.88	15.79	91.63
	3	653.01	538	651.76	809.52	327.40	92.20	8.91	89.42
	5	695.62	557	683.18	820.47	110.20	75.51	12.75	91.33
	10	642.72	506	608.80	853.17	1088.80	159.51	2.43	81.69
	20	671.21	489	611.33	906.03	1927.20	247.33	-1.01	71.61
	30	661.97	503	598.32	920.00	3603.60	423.41	1.82	51.39
	50	674.37	501	598.58	938.66	4418.00	521.15	1.42	40.17
LAMARCK	**Infinite**	**646.57**	**454**	**531.59**	**931.41**	**3261.80**	**421.14**	-	-
	0	695.58	562	695.03	774.38	20.25	65.53	23.79	84.44
	1	700.62	565	701.01	785.68	37.80	66.15	24.45	84.29
	3	667.34	506	662.96	792.90	46.20	67.17	11.45	84.05
	5	710.37	545	706.93	804.12	27.80	67.65	16.70	83.94
	10	683.94	520	663.88	840.19	68.20	70.82	14.54	83.18
	20	641.00	503	605.92	853.68	248.20	85.94	10.79	79.59
	30	657.01	495	616.19	867.08	337.80	98.20	9.03	76.68
	50	641.57	475	576.63	893.78	1021.80	165.98	4.63	60.59

Table 6.35 –Influence of evolutionary theory, Kempe and deviation rule on solution quality for HEC.

Run	% Dev Run Min	Av	Best	Best 10	Worst 10	Eligible	Time	% Dev Best	% Dev Time
DARWIN	Infinite	1030.19	823	979.97	1407.18	6283.00	2184.36	-	-
BALDWIN	Infinite	1042.37	829	929.41	1394.96	6546.20	3532.64	-	-
	0	1084.09	888	1090.28	1168.55	13.00	332.62	7.11	90.38
	1	1055.41	874	1067.81	1201.71	37.80	342.08	5.42	90.32
	3	1039.56	888	1054.76	1224.14	90.00	363.25	7.11	89.72
	5	1006.29	845	1004.09	1241.28	269.40	432.19	1.93	87.77
	10	1027.37	852	1000.00	1319.48	1698.60	1057.29	2.78	70.07
	20	1036.50	811	978.24	1377.27	4774.40	2523.30	-2.17	28.57
	30	1052.07	841	978.11	1423.17	5891.20	3237.96	1.45	8.34
	50	1045.14	838	970.55	1423.73	6557.40	3538.35	1.09	0.16
LAMARCK	**Infinite**	**976.42**	**736**	**858.47**	**1396.81**	**2912.20**	**1524.33**	-	-
	0	1066.71	837	1069.66	1186.51	20.40	301.12	13.72	80.25
	1	1057.73	816	1066.55	1197.31	27.20	303.86	10.87	80.07
	3	1064.18	858	1076.45	1217.16	36.20	306.17	16.58	79.91
	5	988.32	795	992.94	1220.21	129.20	331.32	8.02	78.27
	10	988.48	800	981.81	1276.58	315.80	387.11	8.70	74.61
	20	980.59	776	936.39	1319.41	858.00	611.58	5.43	59.88
	30	955.18	739	906.14	1350.51	1913.80	951.15	0.41	37.60
	50	982.48	782	909.05	1399.43	2398.00	1275.46	6.25	16.33

Table 6.36 –Influence of evolutionary theory, Kempe and deviation rule on solution quality for EAR.

Run	% Dev Run Min	Av	Best	Best 10	Worst 10	Eligible	Time	% Dev Best	% Dev Time
DARWIN	Infinite	1050.06	839	994.44	1409.89	7145.00	4958.45	-	-
BALDWIN	Infinite	1059.28	857	947.51	1381.40	6864	7563.71	-	-
	0	1111.88	913	1116.15	1218.96	19.60	522.93	6.53	93.09
	1	1080.25	917	1095.90	1227.89	32.20	531.61	7.00	92.97
	3	1055.13	902	1066.65	1262.81	94.00	576.41	5.25	92.38
	5	1043.67	854	1042.54	1291.01	177.80	644.79	-0.40	91.48
	10	1035.63	854	1008.28	1329.88	812.60	1164.20	-0.40	84.61
	20	1053.56	852	994.36	1390.88	3720.40	4139.90	-0.60	45.27
	30	1053.83	833	986.04	1402.37	5778.60	5996.73	-2.80	20.72
	50	1057.06	836	982.51	1418.41	6933.40	8497.79	-2.45	12.55
LAMARCK	Infinite	1059.86	831	930.98	1387.76	3688.20	7643.01	-	-
	0	1138.21	928	1139.33	1203.10	13.80	802.47	11.67	89.50
	1	1108.41	862	1114.41	1225.05	20.60	815.15	3.73	89.30
	3	1094.20	915	1105.33	1260.53	36.20	839.49	10.11	89.00
	5	1083.69	864	1079.85	1292.92	63.00	885.29	3.97	88.42
	10	1051.30	856	1027.72	1323.83	218.80	1115.52	3.00	85.41
	20	1057.06	848	1004.39	1376.52	954.40	2382.71	2.05	68.83
	30	1034.76	827	964.39	1403.54	2710.20	5305.71	-6.48	30.58
	50	**1057.18**	**812**	**973.72**	**1421.09**	**3472.20**	**7020.54**	**-2.25**	**8.14**

Table 6.37 –Influence of evolutionary theory, Kempe and deviation rule on solution quality for TRENT.

Results indicate that limiting the use of Kempe chain application does not have a detrimental effect on solution quality. Of the three theories, Lamarckism appears the superior (as highlighted by the *Best* solutions that are bolded). Additionally, the number of timetables eligible for Kempe improvement is drastically reduced through this evolutionary theory. This can be attributed to the wider search processes (see Section 6.6 for further discussion). In consequence, the number of feasible timetables is reduced and, from these feasible solutions, a greater range of ant-based second-order scores is experienced. With regards to the latter point, ant searches are capable of producing a subset of very good solutions. In terms of the decision to improve an ant-based solution via Kempe chains, any future ant based solutions must be of relatively first-class quality to qualify for improvement. These factors contribute to a lower number of timetables that are permitted Kempe based improvement, leading to quicker runtimes. Table 6.38 emphasises some of the above points. Please note the results associated with HEC and EAR. In both instances, the *Av* and *Worst 10* statistics are inferior for Lamarck than Darwin but the *Best* and *Best 10* results are superior. This highlights the comment regarding the distribution of ant-based second-order scores and the standard deviation scores that are presented confirm this.

Returning to Tables 6.35-6.37, it appears that a general deviation rate of 30% is a reasonable compromise between solution quality and computational expense. It has already

been discussed that Lamarckian theory is superior and it is observed that at 30% deviation, great computational effort is saved in return for minimal reduction in solution quality (when compared with infinite deviation). The evidence is as follows. For HEC, there is a 9.03% lowering in solution quality in return for a time saving of 76.68%. For EAR, there is a minimal 0.41% difference in the respective *Best* solutions in return for a runtime decrease of 37.60%. For TRENT, there is an improvement in solution quality by 6.48% and a reduction in computational effort of 30.58%.

Data Set	Strategy	Av	Best	Best 10	Worst 10	Feasible	Std Dev
HEC	*Darwin*	935.66	673	711.28	2237.94	76.84	195.71
	Baldwin	1381.77	785	888.12	2514.64	71.18	260.94
	Lamarck	1384.57	546	659.50	2660.68	32.62	361.76
EAR	*Darwin*	1296.94	1037	1104.02	1788.46	62.85	104.66
	Baldwin	1432.94	1046	1201.00	1819.52	65.46	86.66
	Lamarck	1305.28	830	927.46	1934.88	29.12	185.08
TRENT	*Darwin*	1259.89	985	1039.48	1801.28	71.45	120.51
	Baldwin	1453.69	1133	1204.66	1818.32	68.64	91.98
	Lamarck	1512.94	1099	1180.14	1920.88	36.88	122.03

Table 6.38 – Statistics describing the influence of evolutionary theories on ant-based searches.

Figures A6.1-A6.3 presented in Appendix 6.6 indicate that extra computational effort is required to generate bigger second-order improvements. Consequently, better ant-based solutions will not only contribute to the generation of improved Kempe solutions but will require less runtime to execute. Therefore, the use of Lamarckism will lead to better solution quality and will require lower computational effort than the other evolution theories.

6.3.2.6 Width of Exploration

By previous study, it has been determined that the incorporation of evolutionary theories (Darwin, Baldwin and Lamarck) bears a strong influence on solution quality. Results suggest that Baldwinian and, in particular, Lamarckian theory enhance the exploratory powers of ant-based searches. Consequently, a greater diversity of solutions is obtained. This has two benefits. Firstly, since good feasible timetables receive a greater reward weighting, the ants tend to favour the allocation of good (with respect to fitness) pairwise exams within the same timeslot. This exploitation characteristic, along with improved diversity, leads to improved solution quality. Secondly, a greater variety of feasible timetables, some of fitter quality, will lead to better *After Add-On* solutions. Figures 6.10-

6.12 illustrate one reason for better solution quality with Lamarckian evolutionary theory. The standard deviation statistic of second-order scores within each cycle is plotted to indicate the level of diversity that is achieved and demonstrate that ants do not converge to the same solution. The Figures depict a common appearance. Lamarckism offers the greatest exploration, followed by Baldwinism and then Darwinism. This ordering corresponds to the quality of best solutions obtained with each of the evolutionary theories.

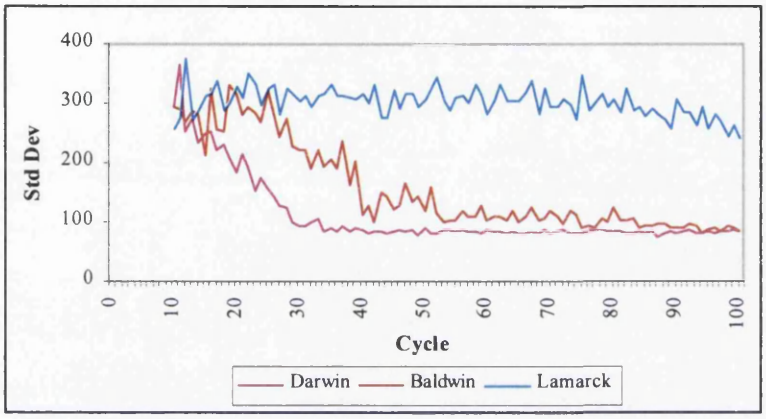


Figure 6.10 – Within cycle standard deviation for evolutionary theories when using Kempe for HEC

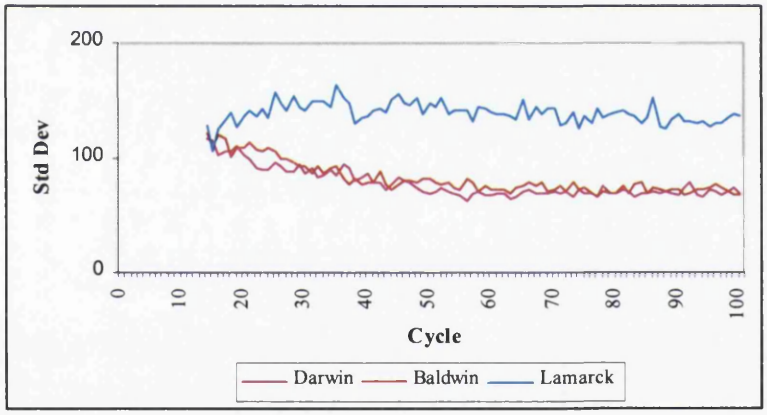


Figure 6.11 – Within cycle standard deviation for evolutionary theories when using Kempe for EAR

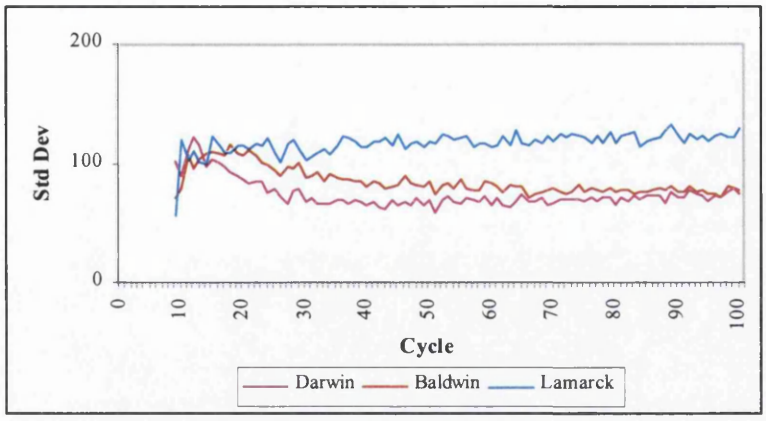


Figure 6.12 – Within cycle standard deviation for evolutionary theories when using Kempe for TRENT

To demonstrate the width of the search, we will examine the mean branching factor $\overline{\lambda}$ - first proposed in Gambardella and Dorigo (1995). The aim of any ant-based search process is to obtain a balance between exploitation and exploration. Stützle and Hoos (1997) suggested that best solutions are obtained in a near stagnation environment since the ants concentrate on a small subset of the solution space. Since $\overline{\lambda}$ indicates the dimension of the solution space that is being investigated, it is claimed by the above authors that lowering $\overline{\lambda}$ to some mark is desirable. The mean branching factor is defined formally in Chapters 2 and 5 but will be revisited here.

Let $AQ_{max}(r,s)$ and $AQ_{min}(r,s)$ be the largest and smallest trail values on all feasible pairwise vertices (r,s) and $d_r=AQ_{max}(r,s) - AQ_{min}(r,s)$. The mean branching factor ($\overline{\lambda}$) for a vertex r is represented by the number of vertices s, with a pairwise trail score greater than λ. $d_r+AQ_{min}(r,s)$, with $\lambda=0.05$. Figures 6.13-6.15 plot the cyclic mean branching factors across the three evolutionary theories for each data set.

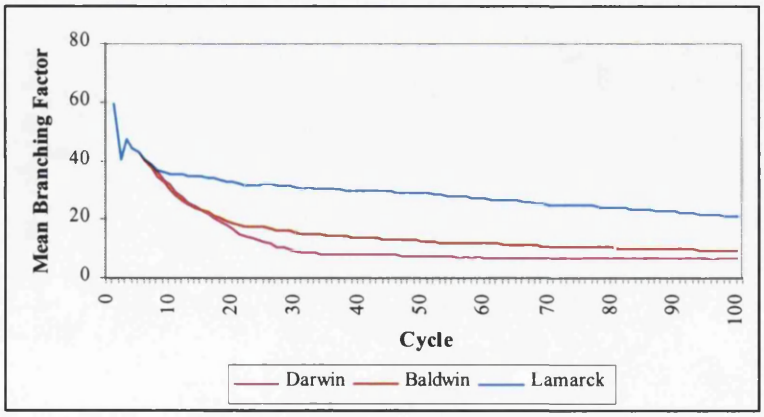


Figure 6.13 – Cyclic mean branching factor for evolutionary theories when using Kempe for HEC

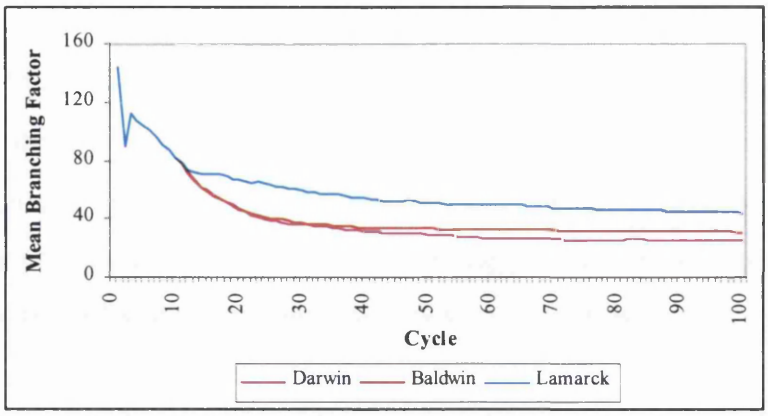


Figure 6.14 – Cyclic mean branching factor for evolutionary theories when using Kempe for EAR.

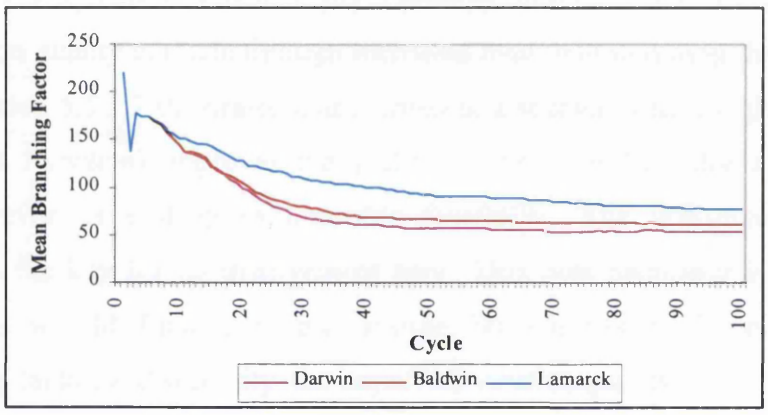


Figure 6.15 – Cyclic mean branching factor for evolutionary theories when using Kempe for TRENT.

Each figure depicts a similar pattern. For each evolutionary theory across each of the data sets, $\overline{\lambda}$ reduces as the search matures. This is a desirable attribute and indicates the search is narrower and consequently, the trails are being exploited, thus (generally) leading to better quality solutions. Stützle and Hoos (1996) demonstrated that better search conditions exist in a near-stagnation state (i.e. $\overline{\lambda} \approx 0$), however the authors imposed a diversification technique that reset the trails when no improvement had been detected after a predetermined number of iterations. Since no diversification strategy is used here, reaching a near stagnation environment too early in the search will limit the exploratory powers of the ants. The benefit of widening the search has already been demonstrated in Section 5.9, which, indicates the benefit of obtaining a search environment that produces values of $\overline{\lambda}$ which suggest that a balance between exploitation and exploration is being achieved.

Lamarckian evolution exhibits a wider search than the other theories and can be attributed to the extending the neighbourhood through Kempe Chains (see Section 6.6 for discussion relating to suitability of Baldwinism for Local Search and Lamarckism for Kempe Chains). Wider search is demonstrated by the predominately larger $\overline{\lambda}$ factor over the cycles and the slower reduction of $\overline{\lambda}$ per cycle. Unquestionably, the use of Lamarckian theory enhances solution quality above the other theories and this suggests that wider search processes are beneficial to the type of search environment imposed on the ants.

It has been determined that the use of Lamarckian theory when used with Kempe Chains maintains good levels of exploration after the infancy of the search. Solution quality improves as a consequence of this diversity. However, in this section, an attempt to further improve solution quality is made through increased exploitation during the latter stages of the search. Section 5.5.3.2 illustrated that increasing a second-order weight d_2 on the ERF (*Elitist Reward Function*) improves the quality of the timetables that are generated. A trade-off, however, is a drop in timetable feasibility. The influence of d_2 on ant performance is the key for its involvement here. This bias parameter is varied and the investigation is twofold. Firstly, we gain insight into the drop in diversity, through the mean branching factor, and secondly, the impact on solution quality.

Figures 6.16 and 6.17 plot the cyclic mean branching factor for a selection of $\overline{\lambda}$ for HEC and EAR respectively. Increasing d_2 does appear to bear influence on diversity in these examples. In Figure 6.16, the narrowest search is demonstrated when d_2=*2000*, while in Figure 6.17, the same setting generates less diversity and consequently, more exploitation characteristics. These settings generate the better *Best* results as produced in Table 6.39.

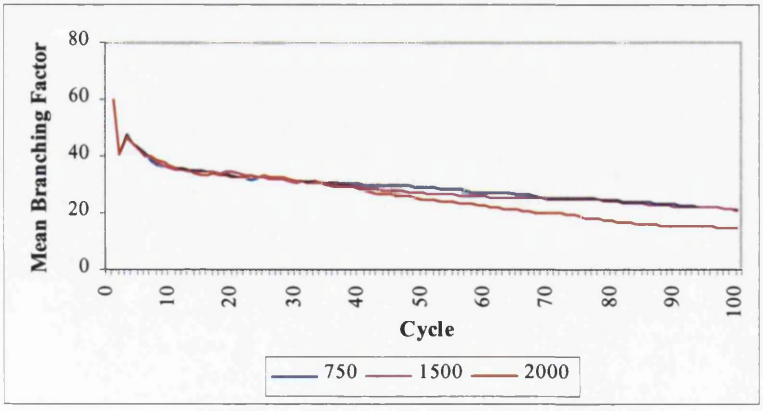


Figure 6.16 – Cyclic mean branching factor for selection of d_2 for HEC.

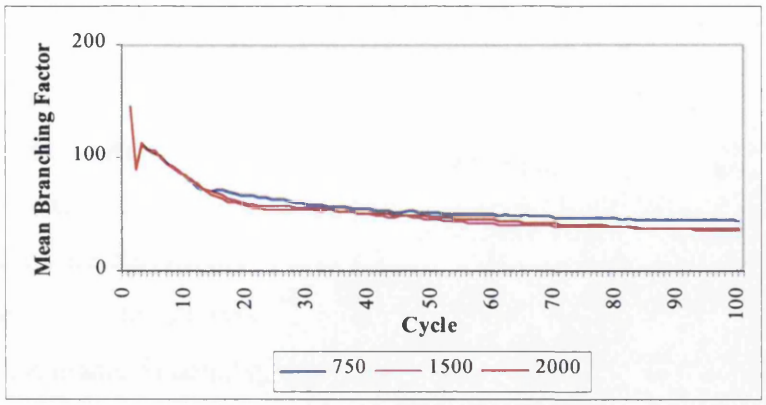


Figure 6.17 – Cyclic mean branching factor for selection of d_2 for EAR.

The mean branching factor plots for TRENT do not show any signs of extra exploitation. The low density of the TRENT graph may be the reason for this. However, the use of higher d_2 does lead to improved solution quality for all data sets. Table 6.39 presents the *Best* statistics for both *Before Kempe* and *After Kempe* solutions. Placing more weighting on d_2 does enhance both *Before Kempe* and *After Kempe* solution quality. The best solutions have been bolded. Surprisingly, results are common across the data sets. Allowing d_2=*2000* generates superior *Before Kempe* results for all three data sets, while similar statements can be made regarding *After Kempe* results, thus emphasising the relationship between these two variables. It can be seen that the disparity between *Best Before Kempe* and *Best After Kempe* solutions is not that large. This suggests that the ants learn (through Lamarckian theory) from the repairs that are made through the *Add-On* strategy (*Kempe Chains*). Additionally, we have determined that Lamarckian strategy encourages search diversity and the number of ant-based solutions eligible for improvement is reduced as a result. Further improvement to *Best Before Kempe* solutions will reduce the volume of eligible solutions even more. Small differences between better *Before Kempe* and *After Kempe* coupled with lower eligibility rates will reduce runtimes further.

	Before Kempe			**After Kempe**		
d_2	*HEC*	*EAR*	*TRENT*	*HEC*	*EAR*	*TRENT*
750	548	779	960	495	739	827
1500	489	744	859	469	726	746
2000	**475**	**731**	**784**	463	**718**	**699**
2500	476	843	803	**461**	785	750
3000	519	841	921	488	757	787

Table 6.39 – Best Before Kempe and After Kempe solutions for experimental d_2 across data sets.

6.4 Comparison across Methods

In this section, we compare some of the methods that have been discussed within Chapter 5 and this current chapter. The aim is to demonstrate the benefit that different theories, enhancement techniques and theoretical modifications can offer with respect to solution quality. Six cases are considered. Firstly, we take the first-order case (labelled M1, Section 5.1), where second-order statistics are just recorded and no improvements with regard to second-order are made. Secondly, the algorithm is modified to accommodate second-order fitness and a second-order bias constant is incorporated (labelled M2, Section 5.4). Thirdly, the *Elitism Reward Function* is used and parameters are set according to the *Robust Elitist Settings* (labelled M3, see Section 5.5.3.2). Fourthly, the hybrid of TSP and Local Search is utilised with both *Before Add-On* and *After Add-On* statistics recorded (labelled M4i and M4ii, Section 6.3.1.6). Fifthly, the graph theoretic Kempe chains is considered (labelled M5i and M5ii for *Before Kempe* and *After Kempe* solutions respectively, Section 6.3.2). Finally, it has been shown that increasing the sample size of chains does increase solution quality. However, the trade-off is the exorbitant runtimes. To compensate for this, an ad-hoc diversification strategy has been incorporated (labelled M6i and M6ii for *Before Kempe* and *After Kempe* solutions respectively). If a new best solution has not been observed for ten cycles then the run is terminated prematurely and a new independent run is started. For each eligible feasible timetable, 5000 Kempe Chains (arbitrary large amount) are sampled. No precursory investigation has been performed for this strategy due to the large runtimes. Both the TSP-Local hybrid and Kempe Chains are used in conjunction with Lamarckian evolution theory and *Before Add-On* solutions are deemed eligible according to the 30% deviation rule. Each of these cases has been applied to members of the data set pool (as described in Chapter 3) and the results are in Table 6.40.

Data Set	*Statistic*	*M1*	*M2*	*M3*	*M4i*	*M4ii*	*M5i*	*M5ii*	*M6i*	*M6ii*
C91	Best	2478	2464	2140	1963	**1361**	2248	1738	2558	1477
	Average	3237.57	3199.99	2847.92	2614.71	1634.14	3176.84	2283.35	3373.55	1807.16
	Feasible	84.06	71.33	62.36	64.84	-	51.04	-	2.90	-
	Time	6771.35	7444.75	7544.00	21489.43	-	69735.20	-	83757.46	-
C92	Best	2495	2442	2031	1787	**1411**	2000	1668	2393	1465
	Average	3441.53	3233.99	2725.30	2629.19	1930.76	3033.05	2075.82	3203.13	1668.56
	Feasible	83.99	46.95	53.93	40.46	-	31.36	-	0.01	-
	Time	3520.71	3920.79	3933.93	10106.54	-	17044.00	-	78594.90	-
EAR	Best	1430	1370	1037	1242	747	779	**739**	1162	694
	Average	1745.50	1715.36	1296.94	1775.80	967.90	1272.95	955.18	1640.61	854.14
	Feasible	76.73	7218	62.83	34.03	-	29.12	-	9.88	-
	Time	188.23	201.47	220.78	510.23	-	951.15	-	1814.32	
HEC	Best	926	723	673	758	467	548	495	741	**442**
	Average	1646.40	1503.46	935.66	1596.00	615.74	1394.39	657.01	1363.83	558.99
	Feasible	87.08	81.15	76.84	52.03	-	32.62	-	12.24	-
	Time	35.69	42.11	42.89	63.20	-	98.20	-	222.07	-
KFU	Best	2162	2094	1616	1994	1135	1426	1079	1715	**921**
	Average	3351.87	3272.10	2415.22	3255.31	1619.88	2603.16	1407.29	2983.73	1150.57
	Feasible	83.70	40.32	63.45	22.89	-	25.54	-	4.61	-
	Time	1671.29	1933.28	1909.32	2442.91	-	4061.30	-	59523.90	-
LSE	Best	863	807	798	921	**501**	719	570	925	508
	Average	1293.39	1238.21	1050.80	1404.88	644.24	1274.25	757.58	1369.87	625.97
	Feasible	78.70	46.99	56.23	31.09	-	15.41	-	0.84	-
	Time	784.55	927.46	932.04	1257.64	-	1867.09	-	5124.18	-
STA	Best	3079	3048	3021	3021	3021	3021	3021	3022	**2975**
	Average	3274.81	3268.45	3120.54	3118.68	3081.52	3148.50	3041.35	3161.78	3026.64
	Feasible	95.63	88.17	94.68	95.28	-	91.82	-	81.40	-
	Time	89.72	112.36	108.42	451.29	-	602.88	-	11173.20	-
TRENT	Best	1255	1215	985	1215	752	960	827	1154	**712**
	Average	1649.83	1578.24	1259.89	1603.03	905.24	1442.89	1034.76	1515.10	863.09
	Feasible	85.90	71.77	71.45	31.19	-	36.58	-	26.30	-
	Time	296.59	301.24	338.71	722.58	-	5305.71		15102.38	
UTA	Best	2424	2281	1962	1889	**1420**	2109	1724	2552	1480
	Average	3219.35	2941.63	2569.74	2646.36	1691.03	2951.58	2123.63	3081.41	1725.98
	Feasible	83.58	65.10	53.48	51.55	-	28.94	-	0.64	-
	Time	5456.48	5992.69	6103.69	18712.89	-	30435.30	-	61475.60	-
UTE	Best	1546	1845	1088	1478	903	628	579	689	**523**
	Average	3099.76	3248.28	2033.06	3201.70	1522.70	2472.14	1055.75	2355.86	816.58
	Feasible	60.77	41.93	70.70	31.09	-	46.31	-	24.76	-
	Time	133.90	164.26	165.77	192.73	-	278.66	-	15603.20	-
YORK	Best	911	875	839	906	680	711	642	818	**596**
	Average	1220.72	1194.89	1056.75	1261.84	846.83	1088.88	804.95	1180.42	718.37
	Feasible	52.77	43.98	60.86	27.74	-	35.14	-	4.97	-
	Time	169.96	184.72	201.74	360.99	-	727.84	-	5449.06	-

Table 6.40 – Comparison of methods for data pool.

A **hyphen (-)** is used in the Feasible and Time rows for M4ii, M5ii and M6ii strategies. These approaches are associated exclusively with M4i, M5i and M6i respectively and consequently, the same *Feasible* and *Time* statistics are recorded. Instead of duplicating results, a hyphen is used.

Best Solutions – Ant-Based Solutions (M1, M2 and M3)

Case M1 is consistently the poorest performer. This is to be expected since no attempt to cater for the second-order problem is included within the construction process. Case M2 does generally produce better solutions than M1, but not on one occasion (UTE). An average improvement of 2.79% is not significant. However, the use of a second-order component within the feedback facility is crucial. There are some M2 results that stand out. A 21.92% improvement is experienced (HEC), but, confusingly, a 19.34% worsening is observed with UTE. The inclusion of elitism (*Robust Elitist Settings*) improves solution quality considerably. The average percentage improvement between M1 and M3 and M2 and M3 are 18% and 15% respectively. Notable improvements are with EAR, KFU and YOR (this data set is dealt with later).

Best Solutions – Improvement Strategies

Before Add-On

The use of smaller sample Kempe chain (M5) leads to higher quality ant-based solutions, while referring to M1, M2 and M3 and is significantly better than M6 (large sample). Populous samples diversifies the search further and also lowers feasibility – both contribute to lower solution quality. An average solution disparity between M4 and M5 of 16.09% exists in favour of the latter case. However, M4 is superior in the instances of C91, C92 and UTA, which coincidently are the larger data sets.

After Add-On

Despite poor ant-based solution quality, M6 generates the superior final solutions. With respect to final solution quality, Case M6 is 10.53% superior to M5 and 6.78% better than M4. It can be argued that large sample Kempe chains could obtain suitable final solutions

from random poor solutions, thus, eliminating the need for good quality ant-based solutions. Also, M4 performs 4.51% superior than M5. This latter result is useful when data sets have limited or no side constraints and illustrates the benefit that can be gained through a less constrained environment.

Feasibility

As expected, M1 generates the greater volume of feasible timetables. M1 has one purpose, to achieve feasibility. The incorporation of second-order functions (M2) has the influence of lowering feasibility rates. In some cases, the drop is fairly significant (see KFU for example). The inclusion of elitism, even with extra second-order bias (parameter d_2) improves first-order capabilities. In three cases (STA, UTE and YOR), the feasibility rate of M3 is better than M1. Due to the first-order replenishment component of elitism, the introduction of Lamarckian theory does not relegate the search process to infeasibility once again. For all data sets, Case M6 lowers feasibility capabilities. This is expected due to the moves across the solution space that are viable through large Kempe samples.

Computational Effort

As expected, the incorporation of any improvement technique leads to increased runtimes. The relative difference between M1, M2 and M3 is small. The use of an *Add-On* bears a significant influence on runtimes. Very large samples require huge effort and consequently, put the observed solution quality into perspective. However, it does show the benefits of increasing sample size. The key is to increase sample size to a level that does not require unrealistic runtimes while still exploiting the relationship between *Before Add-On* and *After Add-On* solutions. Additionally, it has been observed that less computational effort is required to improve better quality ant-based solutions. It is demonstrated that the TSP-Local hybrid generates relatively excellent solutions, while requiring less runtime than Kempe. As stated previously, this hybrid acts as an attractive alternative when limited side constraints have to be observed.

It has been demonstrated that an improvement phase is imperative to solution quality. It has also been observed that the best final solutions are generated from large sample Lamarckian Kempe Chains, however the runtimes are excessive. Meanwhile, TSP-Local

does perform marginally better than smaller sample Kempe in smaller runtimes. However, it is interesting to note the solution quality performance of these strategies when data sets (Swan2000 and Swan2002) with side constraints are considered. The 30% deviation rule is applied in each case. Runtimes are not available since experiments were performed in parallel and efficiencies between machines make computational efforts incomparable.

	Add-on	Average	Minimum
SWAN2000	*Local*	190.15	98
	TSP-Local	143.26	68
	Kempe	100.90	54
SWAN2002	*Local*	273.89	224
	TSP-Local	250.32	187
	Kempe	199.93	126

Table 6.41 – Add-on results for SWAN2000 and SWAN2002

Based on solution quality alone, it is suggested here that Lamarckian Kempe with 30% deviation (Case M5) is the most suitable method. However, even when side constraints are imposed, the results achieved through the TSP-Local hybrid are more attractive.

6.4.1 Comparison of Exam Exchange Methods

Both Local Search and Kempe Chain Descent are exchange methods that move exams to different timeslots if an improvement in solution quality is possible while maintaining feasibility. It has been observed that Kempe (generally) outperforms Local Search and reasoning can be associated with the neighbourhood structure since extra solutions are accessible. However, through comparison of Tables 6.29-6.31 and Tables 6.35-6.37, it can be seen that Steepest Descent returns very competitive results. The *After Loc* statistics are superior to the *After Kempe* statistics for TRENT. The good performance of Steepest Descent is based upon the efficiency of the ant search. However, despite these promising results, Kempe Chain descents, by nature, has the greatest potential to succeed. A Kempe chain neighbourhood allows the searching of a greater part of the solution space around the ant-based solution (starting solution). Consequently, a greater variety of solutions are investigated and the result is better *After Add-On* quality due to improved solution accessibility. The following sequence of Figures plots within cycle standard deviation (of second-order scores of feasible timetables) against cycle number while comparing Kempe Chains against Local Search. It is demonstrated that a constant level of solution variability is present as the search matures while using Kempe chains, which is a desirable

characteristic. Conversely, Local Search feedback actually limits ant searches. Intuitively, this can be explained as follows. During the early stages of the run, the ant searches lack focus, however, as the search matures the ants learn and exploit small areas of the solution space. Consequently, ants return not too dissimilar solutions. The application of Local Search will further enhance similarities between solutions due to the limited neighbourhood structure. The characteristics of these identical or near identical solutions are passed to the trail, thus limiting the foraging capabilities of future ants. The variation that exists can be associated with the feasible timetables that are not improved (deviation criteria) and the infeasible solutions. The scales of *Std Dev* on Figures 6.18-6.20 can be attributed to the average number of students per exam (see Section 5.6.1 for related discussion).

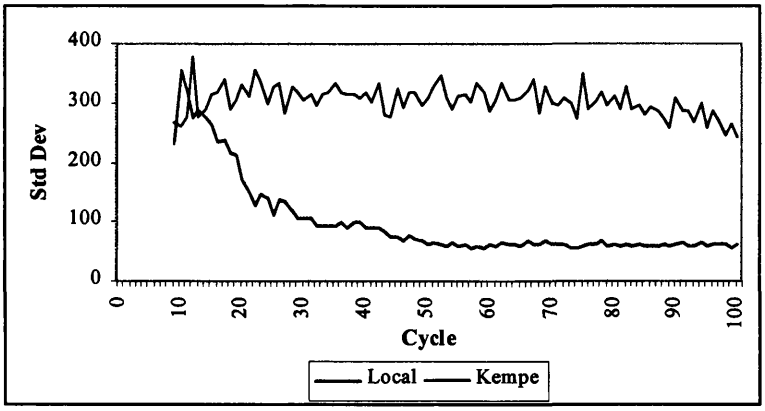


Figure 6.18 – Within cyclic Standard Deviation comparing Local Search and Kempe for HEC.

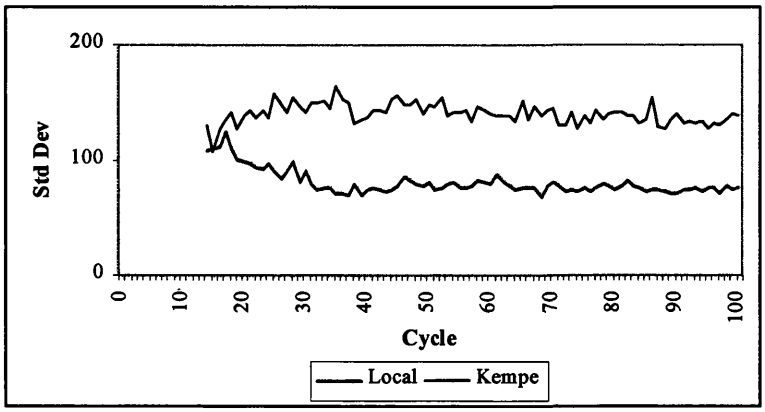


Figure 6.19 – Within cyclic Standard Deviation comparing Local Search and Kempe for EAR.

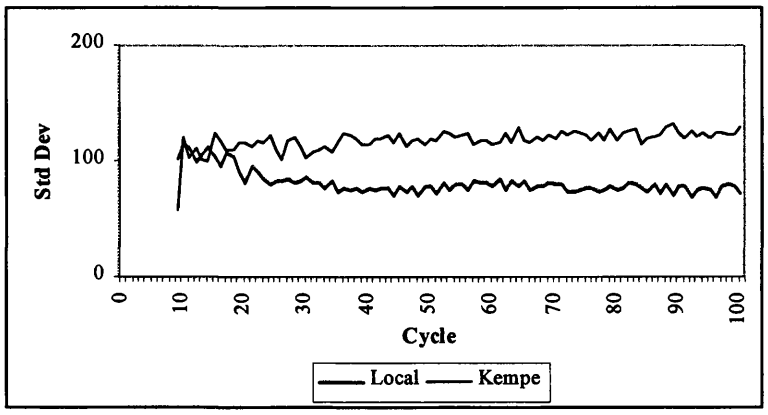


Figure 6.20 – Within cyclic Standard Deviation comparing Local Search and Kempe for TRENT.

For further indication of the extra exploratory power that Kempe Chains offers, the frequency of vertex colour changes per ant between the *Before Add-On* and *After Add-On* solution has been computed for both Kempe Chains and Steepest Descent Local Search. Larger frequencies suggest a greater ability to search larger areas of the solution space around the starting solution (ant-based solution), thus, making better *After Add-On* solutions more accessible. The statistic in question is averaged for each feasible timetable that is deemed eligible for improvement over five independent runs.

	HEC	EAR	TRENT
Kempe	41.48	100.75	148.21
Local	7.93	31.64	49.89

Table 6.42 – Number of vertex colour changes between ant-based and final solution per eligible ant

These statistics indicate that a greater range of solutions (originating from an ant-based starting solution) is obtainable through Kempe, which emphasises the wider neighbourhood structure that is associated with this strategy. The disparities between data sets is linked to the size of data set.

6.4.2 Comparison with Simulated Annealing

It has been demonstrated that solution quality can be drastically improved via a series of 'second-order improving' methods. However, comparison against an alternative optimisation technique is required here to enable overall conclusions regarding the success of the application to be made and to determine whether HAS-EXAM is a valid examination scheduling method. It has been recognised that the use of Kempe Chains (with

Lamarckian evolutionary theory) enhances both *Before Add-On* and *After Add-On*. Kempe Chains played a pivotal role in Thompson (1995) when applying Simulated Annealing to the Examination Timetabling and consequently, seemed a sensible choice for a comparison method in this section.

The method

Simulated Annealing is an improved local search technique, which searches a wider part of the solution space. The analogy of the method stems from statistical thermodynamics, Metropolis et al. (1953), and was recognised as a basis for an optimisation technique, Kirkpatrick et al. (1983) and Cěrny (1985). When applying Simulated Annealing, sequential movements are made across the solution space via small alterations to solutions. Moves are accepted automatically if an improved solution is observed, but acceptance of poorer solutions is dependent on a probabilistic function. More detail regarding this technique can be found in Appendix 6.7.

Kempe Chains with Lamarckian evolutionary theory and the 30% deviation rule is used when applying ants. For SA, a starting temperature of *20* and a cooling ratio of *0.99* are used, with *1000 iterations* at each temperature and a finishing temperature of less than *0.01*. Comparative *Best* statistics are presented in Table 6.43.

Data Set	Ant Best	Simulated Annealing Best
C91	1738	2660
C92	1668	2315
EAR	739	1182
HEC	495	635
KFU	1079	1746
LSE	570	970
STA	3021	2560
TRE	827	990
UTA	1724	2440
UTE	579	1648
YOR	642	1000

Table 6.43 – Statistics comparing SA and AS-EXAM second-order scores

It can be observed in Table 6.43 that the use of ants can lead to very competitive solutions and, with respect to this comparison, outperform established methods. Regrettably, runtimes are not available for SA and a full evaluation of algorithm efficiency cannot be assessed here. However, the quality of solution alone is encouraging. The main advantage

of the ant approach is the manner in which the solution space is explored. The probabilistic nature of the algorithm encourages greater diversity and explorations that are biased will lead to the production of 'fit' timetables. Consequently, the superiority in solution quality over SA can be attributed to the additional diversity of search.

6.5 Justification of Parameter Settings

The work performed in Chapters 5 and 6 has used the parameter settings ($\alpha=2$, $\beta=1$, $\rho=0.5$) that were determined in Chapter 3 and, to a lesser extent, in Chapter 4. However, it is perceivable that different values of α and β may have been appropriate for different investigations, but addressing such sensitivities for each method would be computationally impossible. Therefore, to justify the use of these parameters, the influence of bias parameters on *Best* solution quality with respect to the final solution method (Lamarckian evolutionary method using Kempe with 30% deviation) is tested. Table 6.44 presents the best *After Kempe* second-order scores for HEC, EAR and TRENT for $\alpha \in \{0, 0.5, 1, 2, 3, 4\}$ and $\beta \in \{1, 2, 3, 4\}$.

	HEC				EAR				TRENT			
α/β	*1*	*2*	*3*	*4*	*1*	*2*	*3*	*4*	*1*	*2*	*3*	*4*
0	-	-	584	522	-	-	-	-	-	1024	936	894
0.5	-	557	584	556	-	-	-	-	-	956	921	861
1	564	511	530	521	-	907	855	874	981	815	814	842
2	495	461	455	511	739	754	794	813	827	714	751	765
3	459	**443**	462	484	743	782	783	783	**718**	729	746	749
4	453	467	454	477	743	**709**	719	773	752	743	735	773

Table 6.44 – Best second-order statistics across α and β for Lamarckian evolutionary method with Kempe at 30% deviation.

A hyphen signifies that no feasible timetables were observed at those parameter settings.

Generally, increasing α does enhance solution quality. However, the influence of α does dampen when $\alpha \geq 2$, which corresponds to what has been observed in Chapters 3 and 4. The impact on solution quality when β is increased is negligible. These findings re-assure that the used parameters are acceptable.

6.6 Conclusion

This chapter has demonstrated that *Add-On* techniques enhance solution quality further (than witnessed in Chapter 5). This is not surprising since it has been recommended that local search is used in conjunction with Genetic Algorithms, Burke et al. (1995c) for example. However, it has been demonstrated that the efficiency of the ant-based searches is vital with respect to *Best* solution quality, thus validating the role of the ants. Three mainstream *Add-On* strategies have been used. It has been shown that all techniques are capable of reducing second-order scores, however it is the graph theoretic Kempe chains that has the greatest capacity to succeed due to its ability to widen the neighbourhood around solutions. It is shown though that better solutions are generated through larger Kempe samples, which does require extra computational effort.

The use of the TSP-ANT algorithm to re-order timeslots showed promise but the method is restricted the accommodation of difficult exams (pre-assigned and time-windowed) which, reduces the flexibility of the TSP approach. Meanwhile, Steepest Descent is relatively computationally efficient but is limited in its application. Searches get trapped in local minima, which increase the probability of returning disappointing results. A hybridisation of the TSP model and local search was shown to be very competitive with respect to solution quality while needing smaller runtimes than Kempe.

Traditional evolutionary concepts were used to vary the feedback mechanism when considering each of the *Add-On* techniques. It was shown that passing information, regarding the modified solution, to the trail is beneficial to the overall solution quality. It has been demonstrated that *Before Add-On* and *After Add-On* have a positive relationship. Since these evolutionary theories lead to improved *Before Add-On* solutions, the *After Add-On* solutions are even more superior in consequence.

It was found that Steepest Descent performed better through Baldwinian evolution and Kempe Chain descents through Lamarckian. The reasoning can be attributed to the following. The neighbourhood definition of Local Search is relatively narrow (in comparison to Kempe Chains) and the fitness landscape exhibits a spiky structure. Therefore, the greater variability associated with Baldwinism widens the basin around the minima and enables the reachability of better solutions. Lamarckian evolution will, by

nature, encourage similar solutions due to the restricted neighbourhood and consequently, limits the options for the ants (in comparison to Baldwinian). Meanwhile, the structure of the landscape with Kempe Chains is flatter and consequently, the neighbourhood structure is far wider. In this instance, the exploitation nature of Lamarckism (i.e. associating the *Add-On* fitness score with the *Add-On* solution) actually leads to greater exploration (see Figures 6.17-6.18). Extending the neighbourhood through Kempe Chains allows for a greater range of *After Add-On* solutions, therefore, there is a wider distribution of pheromones across solution characteristics, thus, giving the ants more options. With respect to Baldwinism, the range of solution characteristics obtained through Kempe Chain descents are not passed to the trail (only the fitness score is) and consequently, this reduces the diversity of the search.

The relationship between *Before Add-On* and *After Add-On* has been exploited to reduce the number of timetables that are modified through an add-on strategy, thus reducing computational effort. In some cases, it is found that better solutions are achievable by lowering the number of *Before Add-On* solutions that are modified since there is a resultant greater differentiation between solutions. The number of modified solutions could be lowered further if low exploration is identified. Reduction in the width of the search will result in the construction of similar solutions of close qualities. These *Before Add-On* timetables will potentially qualify for repair and consequently, a series of similar *After Add-On* solutions are produced. This leads to the unnecessary inflation of runtimes. Therefore, it is proposed that an *Add-On* is used infrequently during periods of narrow exploration. This may be suitable for Steepest Descent, however, there is enough evidence of good exploratory width for the Lamarckian Kempe system (as illustrated in Figures 6.13-6.15), thus a range of timetables is constructed at all stages of the search. Therefore, it is felt that limiting the timetables that are modified, even at mature stages of the search, is unjustified.

In addition to reducing the number of timetables eligible for improvement, it is possible for the user to control runtimes. The number of Kempe chain descents can be limited and/or the experimental run could be stopped earlier than pre-specified and the user could still expect a relatively good solution as opposed to SA for example, which tends to generate good solutions at the end of a run when temperatures are lower. Another option is to allow

the run to be performed overnight, which is of limited inconvenience to the user given that institutions only need to construct timetables two to three times annually.

It is observed that (potentially) different bias parameter values are appropriate for different problem types. Typically, within this thesis, it has been shown that $\alpha \geq 2$ is required to generate good solution quality (see Chapters 3 and 4) for the examination timetabling problem but setting $\alpha = 1$ is sufficient for the TSP. This evidence shows the need for some precursory investigation for new problem areas or the introduction of some decision-making system that deduces α according to the search environment, Ross et al. (1998) and White et al. (1997). Applying a standard trail bias parameter to all problem types could lead to poor solution quality.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

The work in this thesis has demonstrated the potential of an ant-based method (HAS-EXAM) to solve the multi-objective examination timetabling problem. HAS-EXAM is particularly effective in dealing with hard constraints and performs well in accommodating the second-order limitations inherent in the problem. In this work, we have introduced methods of enhancement to improve solution quality further (with respect to hard and soft constraints) and have provided a solution method framework that will return consistently good results.

It has been shown that the balance between exploitation and exploration is the key to solution quality. Too much exploration limits the focus of the search, while too much exploitation reduces the area of the solution space that is investigated, thus similar solution copies are constructed and the remainder of the experimental run becomes obsolete.

The HAS-EXAM algorithm is a two-phase approach where the first phase allows the ants to construct solutions. The role of the second phase is to improve the second-order condition of the timetable through a repair facility (known as an *Add-On*). The success of the second phase is based on the neighbourhood definition around the starting solution (ant-based solution) and the volume of repaired solution information that is incorporated within the trail. It is demonstrated that a wider neighbourhood leads to superior solution quality during the second phase. It is shown that the Kempe chain neighbourhood is particularly suited to phase two due to its ability of moving from one feasible solution to the next.

The work described in this thesis has outlined a general solution approach to the examination timetabling problem which allows the incorporation of a wide range of constraints and objectives. The HAS-EXAM algorithm could be developed into user-friendly software. It has been demonstrated that aspects such as room constraints and time-windowed exams are dealt with satisfactorily. The disadvantage of HAS-EXAM lies with

the required computational expense. Runtimes can be minimised through sensible parameter settings [i.e. α *(trail)*, β *(desirability)*, N_A *(number of ants per cycle)* and N_C *(number of cycles per run)*], conservative *Add-On* application and the implementation of runtime conservation methods. Additionally, the scheduler can terminate runs before the completion of N_C and normally obtain good quality solutions. It has been observed that HAS-EXAM is adept at achieving competitive solutions in relatively quick times. However, since the examination timetabling problem is generally solved on only a few occasions per year, the computational requirements (within reason) should not be a valid drawback. In instances where hours (with larger data sets) of runtime are required, overnight experiments can be performed to the limited inconvenience of the scheduler.

Chapter 3 laid the foundations for future investigations in this thesis. It was shown that the construction heuristic and the bias power exponents had a large influence on solution quality. Construction heuristic *C* (Recursive Largest Degree First heuristic) was consistently superior across the test data sets and also stipulating $\alpha \geq 2$ (where α is the trail factor bias) led to the same conclusion. It is seen that the influence of β (desirability bias) is vital for small α, but the role of β is reduced as α increases. It was also observed that higher power components inflate computational effort. Therefore, it was reasoned that $\alpha=2$ and $\beta=1$ was appropriate and these settings were applicable to all construction heuristics under consideration in Chapter 3.

It was felt wise to determine means of conserving runtimes in order to allow for the processing of more experiments and to make ant methods more attractive as a scheduling tool. It was demonstrated that the use of Linear Interpolation (*LI*), to predict the trail and/or desirability factors when raised to a power exponent, not only reduces computational effort but also maintains feasibility rates. Experiments also showed that runtimes could be reduced through limiting the options at each exam-timeslot decision stage through Candidate Lists. It was shown that the use of particular list constitutions increases feasibility but runtime conservation is not as great as for *LI*.

It is shown in Chapter 3 that the basic ANTCOL algorithm, Costa and Hertz (1997), can be used for scheduling and produces very good results. ANTCOL comprehensively outperforms the graph colouring heuristics discussed within this thesis and optimal results

were returned for some of the test data sets (HEC and TRENT). Additionally, near optimum solutions *(optimum+1)* are produced for the other test data set, EAR. In all instances, the *Best* solutions were better than the number of timeslots stipulated for each data set, indicating that ANTCOL has the potential to be the basis of a capable examination timetabling solution method.

Other parameters of interest are the number of ants per cycle (N_A) and the number of cycles (N_C) per experimental run. It was demonstrated that setting N_A approximately equal to the number of vertices in the problem, Dorigo et al. (1991), is not always required. Lower N_A can achieve the optimum result in some instances, however this is associated with easier data sets. If a scheduler wishes to limit iterations, it is demonstrated that a bias towards larger N_C is more beneficial to overall solution quality due to the greater number of trail updates. However, limits to the success of this do exist e.g. when N_C is large and N_A is too small.

An interesting study in Chapter 3 was mixing the construction heuristics that are used per cycle. Typically, if a number n ants use one construction heuristic then (N_A-n) ants use another. It was shown that mixing construction heuristics C and D is particularly successful. Reasoning was attributed to the use (in moderation) of a more deterministic type heuristic (D).

Chapter 4 showed that modifications to the graph allowed the incorporation of important examination timetabling objectives. An additional set of vertices was added to the graph to represent the timeslots, as described in Balakrishnan (1991). Consequently, the insertion of edges would symbolise whether an exam was time-windowed within a set of timeslots, or pre-assigned to a particular timeslot. Additionally, sets of simultaneous exams were merged into one supervertex. All edges incident to all simultaneous exams were also incident to the supervertex. The manner in which these constrained exams can be dealt with in the graph rather than the algorithm itself simplifies the problem. The extra constraints can be incorporated into the underlying model, thus fundamental changes to the algorithm are not required.

It was shown that standardising the reward function improves quality further. This was reasoned by the observation that a problem with a larger optimum number of colours does not differentiate between solutions as greatly as one with a lower optimum.

To fully evaluate ANTCOL as an examination timetabling solution method on a first-order level, two cases were considered. The first was the *Non-Determined Timeslot Case* that attempted to minimise the number of timeslots, while in the second case, the *Determined Timeslot Case*, the number of timeslots was fixed. It was demonstrated that, in both cases, the search process benefited from enhancement techniques. It was shown that an enhanced ANTCOL (AS-EXAM) obtained the optimum solution for 11 (out of 12) data sets and it was observed that these results were very competitive with the benchmark methods. When the timeslots were fixed, the results were again very good. However, the inclusion of seating capacities constrained the problem further. The introduction of a second trail, known as *PET*, which forces the accommodation of difficult exams earlier in the construction process improved solution qualities substantially. It is shown that *PET* outperforms *Hill* and stresses the benefit that can result from a more efficient construction process rather than a repair facility. It is also demonstrated that *PET* requires very little extra computational effort, unlike *Hill*.

Chapter 4 also proved that time-windowed and pre-assigned exams are easily dealt with. Tests performed on the Swansea data sets illustrate this.

Chapter 5 addressed the second-order problem and in response, the order of timeslot construction was altered to favour second-order and the reward function was modified to allow feedback of second-order characteristics. The influence on second-order quality was immediate. Chapter 5 showed that trail intensification bears a large influence on solution quality. A series of methods were tested and it was demonstrated that elitism was the superior approach. An elite list with membership subject to the *Elitist Reward Function (ERF)* not only maintains feasibility but places importance (through a second-order bias weight δ_2) on second-order characteristics. Increasing this weight does reduce the second-order conflicts per timeslot, but on a downside, lowers feasibility. The trails associated with the elite solutions were also magnified by a weight σ. In practice, this reduces the number of ants that contribute *influential* feedback to the trail.

Chapter 5 demonstrated that, despite extra trail intensities, stagnation is not reached and some exploration is still being performed towards the latter end of the search. It is shown that the success of the elitist method is associated with the more controlled feedback passed on during the early phase of the search. More focus during the infancy of a search leads subsequent ants into more beneficial areas of the solution space.

Extra advancements are firstly, presented through a weighted construction heuristic that encourages the dealing of higher clash exams earlier in construction process and secondly, imposing upper trail limits allows for greater exploration.

Chapter 5 regarded the use of a second trail that was issued with different search objectives e.g. construct timetables with the objective of minimising second-order conflict exclusively. Two studies were performed which differed in the way the trails were updated. The first study allowed for two populations that searched the solution space in a near-independent manner. At strategic points during the search, the second population injects information into the first population. Despite the potential of this approach, it was found that providing the populations with different search objectives does not work as well as using the *ERF* with both populations. This can be attributed to the efficiency of the *ERF* as a feedback mechanism and improved solutions over the one-trail system can be associated with the exploitation of the inherent randomisation in the algorithm. The second study allowed only one population but changed the update mechanism of the second trail to replenish the trails of pairwise exams in adjacent timeslots. This information was represented at each decision stage through a second trail factor that was incorporated into the RPR. It was shown that better results were achievable but not significantly superior and this can be associated with the following reasoning. It was demonstrated that a significant positive relationship exists between first and second trail factors and, in many cases, these scores are related with respect to rank. As a result, the decision of which exam to insert according to the first trail factor would generally correspond to the second trail factor. Therefore, only on rare occasions does a relatively strong second trail factor score lead to a different exam insertion that would normally be performed under the one-trail system.

The aim of Chapter 6 was to introduce means of improving solution quality further. This was managed through the use of an *Add-On* strategy and the exploitation of evolutionary theory. It is found that the general best improvement technique is Kempe chains and this

can be attributed to the greater reachability of solutions from a given starting solution (ant-based solution). However, searches using Kempe chain neighbourhoods are relatively computationally demanding and it has been shown that comparatively good solutions can be found relatively quickly through TSP and in particular, local search. Additionally, a TSP and local search hybrid (TSP-Local) is found to be very competitive with respect to solution quality while requiring lower runtime than Kempe chains. However, the disadvantage of TSP-Local rests with its unsuitability for institutions that impose some other student comfort criteria. Typically, if an institution wishes to minimise the number of students that sit consecutive exams during the *same day*, the TSP becomes no longer suitable (Chapter 2 details examples of this). The Local Search strategy can easily be adapted to such student criteria but is inherently (see Section 6.4.1) inferior to Kempe chains with respect to solution quality and this can be attributed to neighbourhood definition. Local searches are performed within narrow areas since the solution space is spiky by nature. Meanwhile, the Kempe chain neighbourhood is flatter and increases the reachability of solutions. It is recommended here that Kempe chains should be used within an ant-based generalised university timetabling solution method.

It is demonstrated that feeding information regarding modified solutions into the trail is beneficial to the ant-based search. Both Baldwinian and Lamarckian philosophies lead to better ant-based solutions than Darwinian. It is shown that these searches maintain a wider neighbourhood that is explored by the ants, while evidence that the searches become narrower (which is a desirable characteristic since exploitation is increased) over time is also presented. It is shown that the nature of Baldwinian theory can lead to production of some volatile results and this can be attributed to the nature of this evolutionary philosophy, which associates the ant-based solution with the modified solution fitness score. However, this volatility is suitable for narrower neighbourhoods. The basin around the minima widens and better quality solutions result. The superiority of Baldwinian over Lamarckian when using Steepest Descent illustrates this. However, Lamarckian performs better when in conjunction with a Kempe Chain neighbourhood. The significant increase in reachability through sampling Kempe chains encourages a greater diversity of information to be passed back to the trail, thus broadening the ant search. Lamarckian evolution used in conjunction with Kempe produces the fittest solutions.

Chapter 6 also exhibits evidence that a strong relationship exists between the fitness scores of the ant-based and modified solutions for all add-ons. This indicated that computational effort could be reduced through improving a proportion of the relatively better ant-based solutions. It was demonstrated that repairing those solutions that were within 30% of the run-best solution was the best compromise between quality and computational effort. It was shown that reducing *Add-On* application actually improves ant-based solution quality further, which can be associated with the resultant additional differentiation between solutions. As a result of improved ant-based solution quality, overall solution quality also improves.

Extra important conclusions that can be drawn from Chapter 6 are as follows. Firstly, It is demonstrated that *Before Add-On* and *After Add-On* are significantly positively related. Therefore, the efficiency of the ant-based system is critical to the quality of the overall best solution. Secondly, fitter ant-based solutions will require less repair time than poorer solutions, thus, reducing overall computational effort and increasing the attractiveness of the method. To mark the incorporation of on *Add-On*, AS-EXAM is redefined as HAS-EXAM (Hybrid Ant System for the Examination Timetabling Problem).

To emphasise the potential of HAS-EXAM as an examination timetabling solution method, a comparison exercise that showed that HAS-EXAM outperformed SA was presented in Chapter 6 and a second comparative study is shown here. Carter et al. (1996) presented an uncapacitated version of the examination-timetabling problem. The number of timeslots was fixed and an objective function was used to space out the students' exams. The objective function applies a weighting scheme that corresponds to the proximity costs that were first presented in Laporte and Desroches (1984). A penalty w_t is applied to a timetable whenever a student has to sit two exams scheduled t timeslots apart, with w_1=16, w_2=8, w_3=4, w_4=2 and w_5=1. A more detailed explanation of these weights can be found in Chapter 2. In Carter et al. (1996), the total penalty is divided by the number of students and will be defined here as *Carter Cost*. The resulting score acts as the fitness value for a given timetable. No differentiation between consecutive exams within the same day and overnight is made.

It should be noted here that the *Carter Cost* was not applied throughout this thesis due to unavailability of benchmark results (see Table 7.1) during the early stages of research.

HAS-EXAM (Lamarckian theory, Kempe, 30% Deviation) is used for this study. However, in order to accommodate this new weighted objective function, some of the parameter settings that have been used in this thesis are no longer applicable due to differences of scale. Typically, it was shown that a robust setting of the weight δ_2 in the *ERF* is 750. However, this parameter setting was finalised with respect to second-order scores and now we are considering different fitness values.

The reward function is now defined as

$$\frac{1}{K+1}+\frac{\delta}{Carter\ Cost} \quad (7.1)$$

and the ERF is now defined as

$$\frac{1}{K+1}+\frac{\delta_{3,d}}{Carter\ Cost} \quad (7.2)$$

where $\delta_{3,d}=\delta+\delta_2$ and refers to the weight placed on the *Carter Cost* for data set *d*. Here, a combined term, rather than two separate terms, δ and δ_2 is examined due to time constraints.

Since time constraints prevent comprehensive precursory investigation, an estimate of δ_2 is required. Merlot et al. (2002) tabulates the *Average* costs for the data sets of interests for four different approaches. This information not only allows insight of the standard of solutions to expect but also hints towards the weights of $\delta_{3,d}$ that should be used. In Chapters 5 and 6, a static $100+\delta_2$ weight was used due to the absence of priori information regarding typical second-order scores for each of the data sets. Even though a robust parameter is desirable and the setting used was suitable for the three test data sets, doubts regarding the setting of $100+\delta_2=750$ generically are warranted due to differences of scale between data sets i.e. second-order scores variations. Section 7.2 discusses the potential of adaptive weights, which would deal with this concern. Please refer to Appendix 7.1 for the actual $\delta_{3,d}$ settings that were used and how these parameter settings were computed.

The methods described in Merlot et al. (2002), Caramia et al. (2000) and Carter et al. (1996) have been described in Chapter 2, while Di Gaspero and Schaerf (2000) and White and Xie (2000) discussed Tabu Search methods for the examination timetabling problem. *Carter Costs* for these methods, along with HAS-EXAM, are presented in Table 7.1.

Data Set	Statistic	HAS-EXAM	Merlot et al.	Carter et al.	Caramia et al.	Di Gaspero & Schaerf	White & Xie
C91	*Best*	6.16	**5.10**	7.10	6.60	6.20	-
	Average	7.16	5.20	8.38	-	6.50	-
C92	*Best*	5.48	**4.30**	6.20	6.00	5.20	-
	Average	6.22	4.40	7.04	-	5.60	4.70
EAR	*Best*	36.32	35.10	36.40	**29.30**	45.70	-
	Average	39.21	35.40	40.92	-	46.70	-
HEC	*Best*	9.55	10.60	10.80	**9.20**	12.40	-
	Average	11.54	10.70	15.04	-	12.60	-
KFU	*Best*	13.84	**13.50**	14.00	13.80	18.00	-
	Average	15.13	14.00	18.76	-	19.50	-
LSE	*Best*	10.62	10.50	10.50	**9.60**	15.50	-
	Average	11.41	11.00	12.36	-	15.90	-
STA	*Best*	**157.03**	157.30	161.50	158.20	160.80	-
	Average	157.11	157.40	167.14	-	166.80	-
TRENT	*Best*	**8.30**	8.40	9.60	9.40	10.00	-
	Average	8.61	8.60	10.78	-	10.50	-
UTA	*Best*	4.16	**3.50**	**3.50**	**3.50**	4.20	-
	Average	4.78	3.60	4.80	-	4.50	4.00
UTE	*Best*	24.97	25.10	25.80	**24.40**	29.00	-
	Average	25.74	25.20	30.78	-	31.30	-
YOR	*Best*	36.33	37.40	41.70	**36.20**	41.00	-
	Average	38.26	37.90	45.60	-	42.10	-

Table 7.1 – Carter Costs for five benchmark methods and HAS-EXAM

The results presented in Table 7.1 demonstrate that the HAS-EXAM algorithm is a very good examination timetabling solution method in comparison with other published methods. Results presented by Caramia et al. (2000) and Merlot et al. (2002) are the forerunners but the results associated with HAS-EXAM are also impressive. It is shown that a new *Best* solution is observed on two occasions (STA and TRENT) with HAS-EXAM. The largest disparity in quality is associated with the EAR data set. Caramia et al. (2000) returned a *Best* cost solution of 29.30, while the HAS-EXAM method managed a *Best* of 36.32. However, HAS-EXAM outperforms two of the benchmark methods for EAR. The runtimes are not presented here but correspond to the times that have been presented in Table 6.40, Column M5i and consequently, in some cases, need hours of computational effort. However, full experimental runs can be achieved in significantly less than one day and guarantee very good solution quality. It should be noted here that

experiments relating to Table 7.1 include timeslot vertices, whereas the other authors did not incorporate these and this will bear an influence on runtimes.

7.2 Future Work

Chapter 2 introduced the term ACO (Ant Colony Optimisation), which referred to a family of ant based algorithms. For example, Stützle and Hoos (1996) and Maniezzo et al. (1998) presented the MMAS and ANTS systems respectively. A comparative analysis of such algorithms, along with ANTCOL, could be performed to assess the suitability of each type of method for the examination timetabling problem.

Chapter 4 introduced the intelligent diversification function *PET* that guides the search away from solutions that fail to accommodate certain vertices within the colouring on a frequent basis. It was stated in Chapter 4 that *PET* could also be applied in the *Non-Determined Timeslot Case* (in addition to *Determined Timeslot Case*) where the vertices in the x highest numbered colours receives additional trail. This has obvious uses in alternative graph colouring modelling type problems.

Chapter 6 demonstrated that solution quality improvements could be made to ant-constructed timetables through *Add-On* techniques. Extending the neighbourhood through Kempe chains has been shown to the most beneficial to overall quality. Since it has been proven that there exists a positive relationship between *Before Add-On* and *After Add-On* scores, it is proposed here that the *Best After Add-On* solution could be treated as a starting solution and further improvement could be made through a more sophisticated local search technique such as Simulated Annealing. Given that the solutions achieved through HAS-EXAM are very good, it is perceivable that limited additional search would improve solution quality further. However, it should be pointed out that such incorporation reduces the overall influence of the ants and given the context of this thesis (*Examination Scheduling Using The Ant System*) lowers the attractiveness of such a collaboration of search methods.

Chapter 6 demonstrated that rearranging timeslots (in an appropriate manner) lowers second-order scores by notable margins. Given the nature of this thesis, it was deemed suitable to use an ant-based TSP to manage the re-organisation of the timeslots. However,

it is suggested here that an exact TSP could also be used given that the number of timeslots in most examination timetabling problems is not excessive. Consequently, exorbitant computational effort would not be required.

In Section 6.4, we compared three *Add-On* procedures (*Local*, *TSP_Local* and *Kempe*) when using the Swansea data sets. However, runtimes were not detailed due to the differences in machine efficiencies (experiments run in parallel). This could be overcome through the introduction of a scaling factor that indicates the level of relative computational efficiency of a machine.

Chapter 5 used elitism to improve solution quality. It was shown that the use of a joint elitist function, known as the *Elitist Reward Function*, performed very well with respect to the overall objective. It was observed that the second-order weight, δ_2, played a significant role on quality of second-order scores, but the trade-off was the reduction in feasibility. Research to obtain a system of adaptive δ_2 was restricted to a superficial level but it is felt that the development of a suitable δ_2 management system would aid solution quality. Chapter 6 proved that higher δ_2 reduces the area of the solution space that is investigated as represented through the mean branching factor (λ), Gambardella and Dorigo (1996). It is proposed here that λ could be used to monitor the use of δ_2. At the start of an experimental run let $\delta_2=g$ (g is a parameter). We observe this parameter setting until λ dampens when $\delta_2=g+f$ (where f is a parameter). In theory, λ should continue to decrease due to the heightened exploitation that is possible through higher δ_2. A disadvantage of this approach will be the overall reduction in exploration that will result through higher δ_2. To counteract this, it is suggested that all trails should be set at τ_{max} (some upper trail level), Stützle and Hoos (1997), if no improvement has been observed for S cycles. This is a diversification tactic and removes the need of numerous independent experimental runs since one run with diversification is adequate.

Static bias parameters (α and β) have been used throughout this thesis. It has been proven that settings of $\alpha=2$ and $\beta=1$ generate very good solution quality. However, favourable conclusions regarding these values were the result of precursory investigation and, to our current knowledge, can only be applied to examination timetabling problems. To develop Ant Algorithms (or HAS-EXAM) into a robust solution method that could be applied to a

plethora of problem types, some intervention of artificial intelligence is required. Burke and Petrovic (2002) and Ross et al. (1998) claim that the future of meta-heuristic solution methods belongs with hyper-heuristics. One heuristic is treated as the management system to decide the appropriate solution method to use given the search environment. The use of hyper-heuristics could be used to decide the appropriate bias parameter settings. White et al. (1997) indicate that Genetic Algorithms could be used to chose appropriate α and β.

For completeness, the HAS-EXAM algorithm could be extended to allocate exams to rooms. A trail that is dimensioned exams by rooms could be introduced to evaluate the quality of solutions obtained by placing an exam i into a room j.

The general framework of HAS-EXAM could be evaluated for other multi-objective scheduling problems. It will be of interest to deduce the adaptability (positive signs are shown through good *Carter Costs* that are generated) and generality of this solution method with respect to other problem types. Main areas of concern would include fitness representation (reward function), trail intensification strategies and the widening of the neighbourhood around the ant-based solution. A typical area of application is the nurse-scheduling problem. The University of Wales Swansea has liased with local hospitals and accumulated much relevant data and applied a variety of solution methods to this problem area. The most recent was a successful study involving Genetic Algorithms, which consequently has generated results that could be used comparatively and therefore, used to test the effectiveness of such an ant-based solution method for any future application.

7.3 Final Conclusion

This thesis has demonstrated that the HAS-EXAM algorithm is a very successful examination timetabling solution method on both first and second-order levels. However, drawbacks do exist. Precursory investigation is required to deduce parameter settings that are instrumental in contributing to overall solution quality. Mis-specification of parameters could lead to poor results. Additionally, exorbitant runtimes can result through inefficient use of the solution method. However, it is concluded that the solution quality that is achievable through HAS-EXAM outweighs these disadvantages.

Bibliography

Ahuja, R.K., Orlin, J.B. and Tiwari, A., A Greedy Genetic Algorithm for the Quadratic Assignment Problem, *Computers and Operational Research* **27**, (2000), pp917-934.

Arani, T., Karwan, M. and Lotfi, V., A Lagrangian Relaxation Approach to Solve the Second Phase of the Exam Scheduling Problem. *Journal of the Operational Research Society*, **34**, (1988), pp372-383.

Arani, T. and Lotfi, V. A Three Phased Approach to Final Exam Scheduling. *IIE Transactions* **21**, No. 1, (1989), pp86-96.

Balakrishnan, N., Examination Scheduling: A Computerized Application. *OMEGA International Journal of Management Science*, **19**, No.1, (1991), pp37-41.

Balakrishnan, N., Lucena, A. and Wong, R.T., Scheduling Examinations to Reduce Second-Order Conflicts. *Computers and Operations Research* **19**, No.5, (1992), pp.353-361.

Baldwin, J. M, A new factor in evolution, American Naturalist, **11**, pp441-451, (1896).

Barham, A. M. and Westwood, J. B., A Simple Heuristic to Facilitate Course Timetabling. *Journal of the Operational Research Society*, **29**, No.11, (1978), pp1055-1060.

Brélaz, D., New Methods to Color the Vertices of a Graph. *Communications of the ACM*, **22**, No.4, (1979), pp251-256.

Broder, S., Final Examination Scheduling. *Communications of the ACM*, **7**, No.8, (1964), pp494-498.

Brown, J., Chromatic Scheduling and the Chromatic Number Problem. *Management Science*, **19**, 4, (1972), pp456-463.

Bull, F. and Fogarty, T.C., An Evolution Strategy and Genetic Algorithm Hybrid: An Initial Implementation and First Results. Evolutionary Computing. *Lecture Notes in Computer Science* **865**, Ed. Fogarty Terence, C., Springer-Verlag, (1994), pp95-102.

Bullnheimer, B., An Examination Scheduling Model to Maximize Students' Study Time. In Proceedings of the Second International Conference on the Practice and Theory of Automated Timetabling. *Lecture Notes in Computer Science* **1408**. Eds. Burke, E.K. and Carter, M.W., (1997a), pp78-91.

Bullnheimer, B., Hartl, R.F. and Strauss, C, Applying the Ant System to the Vehicle Routing Problem. Proceedings from the 2nd International Conference on Metaheuristics - MIC97, (1997b).

Bullnheimer, B., Hartl, R.F., and Strauss, C., An improved ant system algorithm for the vehicle routing problem. Technical Report POM - 10/97, Institute of Management Science, University of Vienna, Austria. (1997c)

Bullnheimer, B., Hartl, R.F. and Strauss, C., A New Rank Based Version of the Ant System, *Central Journal for Operations Research and Economics*, **7**, No. 1, (1999), pp25-38.

Bullnheimer, B., Kotsis, G. and Strauss, C., Parallelization Strategies for the Ant System. Technical Report POM 9 - 97 University of Vienna, Austria (1997d)

Burke, E.K., Elliman, D.G. and Weare, R., A Genetic Algorithm Based University Timetabling System. In Proceedings of the Second East-West International Conference on Computer Technologies in Education **1**, (1994a), pp35-40.

Burke, E.K., Elliman, D.G. and Weare, R., A University Timetabling System Based on Graph Colouring and Constraint Manipulation. *Journal of Research on Computing in Education* 27, No.1, (1994b), pp1-18.

Burke, E.K., Elliman, D., Ford, P. and Weare, R., Examination Timetabling in British Universities - A Survey. Proceedings of the First International Conference on the Practice and Theory of Automated Timetabling, *Lecture Notes in Computer Science* **1153**. Eds. Burke, E.K. and Ross.P., (1995a), pp76-92.

Burke, E.K., Elliman, D.G., and Weare, R.F., A Hybrid Genetic for Highly Constrained Timetabling Problems. In Proceedings of the Sixth International Conference on Genetic Algorithms, Ed. Eshelman, L.J. Morgan Kaufmann, (1995b), pp605-610.

Burke, E.K., Newall, J.P. and Weare, R.F. A Memetic Algorithm for University Exam Timetabling. Proceedings of the First International Conference on the Practice and Theory of Automated Timetabling. *Lecture Notes in Computer Science* **1153**, Eds. Burke, E. and Ross, P. Springer-Verlag, (1995c), pp241-250.

Burke, E. K., Elliman, D. and Weare, R., Specialised Recombinative Operators for Timetabling Problems. Proceedings of the AISB Workshop, Lecture Notes in Computer Science **993**. Ed. Fogarty, T.C. Springer-Verlag, (1995d), pp75-85.

Burke, E.K., Newall, J.P. and Weare, R.F., A Simple Heuristically Guided Search for the Timetable Problem. Proceeding of the International ICSC Symposium on Engineering of Intelligent Systems (EIS'98). Eds. Alpaydic. E.,Fyte. C., University La Laguna. (1998). pp574-579

Burke. E.K., Bykov, Y. and Petrovic, S., A Multicriteria Approach to Examination Timetabling. Practice and Theory of Automated Timetabling III. *Lecture Notes In Computer Science* 2079, Eds. Burke, E. and Erben, W. Springer-Verlag, (2000), pp.118-131.

Burke, E.K. and Petrovic, S., Recent Research Directions in Automated Timetabling. *European Journal of Operational Research*, 140, No.2, (2002), pp266-280.

Ĉangalović, M. and Schreuder, J.A.M., Exact Colouring Algorithm for Weighted Graphs Applied to Timetabling Problems with Lectures of Different Lengths. *European Journal of Operational Research*, **51**, (1991), pp248-258.

Caramia, M., Dell'Olmo, P., Italiano, G.F., New Algorithms for Examination Timetabling, Alogrithm Engineering. *Lecture Notes in Computer Science* **1982**, Ed. Naher S. and Wagner D. Springer-Verlag, Berlin, Heidelberg, (2000), pp230-241.

Carter, M.W., A Survey of Practical Applications of Examination Timetabling Algorithms. *Operations Research*, **34**, No.2, (1986), pp193-202.

Carter, M.W., A Lagrangian Relaxation approach to the classroom assignment problem. *INFOR* , **27**, No.2, (1989), pp230-246.

Carter, M.W., Laporte, G. and Chinneck, J.W., a General Examination Scheduling System. *INTERFACES* , **24**, No.3, (1994), pp109-120.

Carter, M.W., Laporte, G. and Lee, S.Y., Examination Timetabling: Algorithmic Strategies and Applications. *Journal of the Operational Research Society*, **47**, (1996), pp373-383.

Cěrny, V., A Thermodynamic Approach to the Travelling Salesman Problem: An efficient Simulation Algorithm. Journal of Optimisation Theory and Applications, **45**, (1985), pp41-55.

Chams, M., Hertz, A. and de Werra, D., Some Experiments with Simulated Annealing for Coloring Graphs. *European Journal of Operational Research*, **32**, (1987), pp260-266.

Chan, P. and White, G. M., An algorithm for examination scheduling: Theory and Practice. Proc. C.I.P.S. Canadian Computer Conference, Edminton (1978), pp244-249.

Cole, A.J., The Preparation of Examination Time-Tables Using a Small-Store Computer. *Computer Journal*, **7**, (1964), pp117-121.

Colijn, A.W. and Layfield. C, Interactive Improvement of Examination Schedules. Proceedings of the ICPTAT 1995 Conference, Edinburgh. Eds. Burke, E.K., Corne, D., Paechter, B. and Ross, P., (1995), pp112-121

Colijn, A.W., Conflict Reduction in Examination Schedules. University of Calgary. Internal Report. (1997).

Colorni, A., Dorigo, M. and Maniezzo, V., Genetic Algorithms and Highly Constrained Problems: The Time-Table Case. Published in: Proceedings of the First International Workshop on Parallel Problem Solving from Nature, Springer, (1990) pp55-59.

Colorni, A., Dorigo, M., Maniezzo, V. and Trubian, M., Ant Systems for Job-Scheduling. *Belgian Journal of Operational Research, Statistics and Computer Science*, **34**, 1, (1994), pp39-53.

Colorni, A., Dorigo, M., Maffioli, F., Maniezzo, V., Righini, G. and Trubian, M., Heuristics from nature for hard combinatorical problems. *International Transactions in OR*, **3**, 1 (1996), pp1-21.

Corne, D., Fang, H. L. and Mellish, C., Solving the Modular Exam Scheduling Problem with Genetic Algorithms. University of Edinburgh, Dept of A.I Research Paper No.622, (1993).

Corne, D., Ross, P. and Fang, H-L., Fast Practical Evolutionary Timetabling. Evolutionary Computing. *Lecture Notes in Computer Science*. **865**, Ed. Fogarty Terence, C. Springer-Verlag, (1994), pp251-263.

Costa, D. and Hertz, A., Ants Can Colour Graphs. *Journal of Operational Research*, **48**, 3, (1997), pp295-305.

Costa, D., Hertz, A. and Dubuis, O., Embedding a Sequential Procedure Within an Evolutionary Algorithm for Coloring Problems in Graphs. *Journal of Heuristics*, **1**, (1995), pp105-128.

Coyne. J, and Paton, R., Genetic Algorithms and Directed Adaptation. Evolutionary Computing. *Lecture Notes in Computer Science*. **865**, Ed. Fogarty Terence, C. Springer-Verlag, (1994), pp103-114.

De Jong, K.A., The Analysis of the Behaviour of a Class of Genetic Adaptive Systems. Doctoral Dissertation, University of Michigan, (1975).

Desroches, S., Laporte, G. and Rousseau, J-M., HOREX: A Computer Program for the Construction of Examination Schedules. *INFOR*, **16**, No.3, (1978), pp294-298

Di Caro, G. and Dorigo, M., AntNet: Distributed Stigmergetic Control for Communications Networks. *Journal of Artificial Intelligence Research*, **9**, (1998), pp317-365.

Di Gaspero, L. and Schaerf, A., Tabu Search Techniques for Examination Timetabling. Practice and Theory of Automated Timetabling III. *Lecture Notes In Computer Science* **2079**, Eds. Burke, E. and Erben, W. Springer-Verlag, (2000), pp.104-117.

Doerner, K., Hartl, R.F. and Reimann, M., Cooperative Ant Colonies for Optimising Resource Allocation in Transportation. *Lecture Notes in Computer Science* **2037**. Eds. Boers, E.J.W., Springer-Verlag, (2001), pp70-79.

Dorigo, M. and Gambardella, L.M., A Study of Some Properties of Ant-Q. Proceedings of PPSN IV-Fourth International Conference on 'Parallel Problem Solving From Nature', Eds. Voigt, H.-M., Ebeling, W., Rechenberg, I. and Schwefel H.-S. Springer-Verlag, (1996), pp656-665.

Dorigo, M. and Gambardella, L.M., Ant Colonies for the Travelling Salesman Problem. *Biosystems*, **43**, 2 ,(1997), pp73-81.

Dorigo, M., Maniezzo, V. and Colorni, A. Positive feedback as a search strategy. Technical report 91-016, Dipartimento di Elettronica Informazione, Politecnico di Milano. (1991).

Dorigo, M., Maniezzo, V. and Colorni, A., The Ant System: Optimisation by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, **26**, No.1 (1996), pp1-13.

Dowsland, K.A., A Timetabling Problem in which Clashes are Inevitable. *Journal of the Operational Research Society*, **41**, 10, (1990), pp907-918.

Dowsland, K.A., Genetic Algorithms - a Tool for OR? *Journal of the Operational Research Society*, **47**, (1996), pp550-561.

Dowsland, K.A., Pugh, N.J. and Thompson, J.M., Examination Timetabling with Ants. In Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling. Abstract. Eds Burke. K. and De Causmaecker. P., (2002), pp397-399.

Dueck, G., New Optimization Heuristics - The Great Deluge Algorithm and the Record-to-Record Travel. *Journal of Computational Physics*, **103**, (1992), pp1-7.

Dunstan, F.D.J., Sequential Colourings of Graphs. *Congressus Numerantium*, **15**, (1976), pp151-158.

Eglese, R.W., Simulated Annealing: A Tool for Operational Research. *European Journal of Operational Research*, **46**, (1990), pp271-281.

Eiselt, H.A. and Laporte, G., Combinatorial Optimization Problems with Soft and Hard Requirements. *Journal of the Operational Research Society*, **38**, 9., (1987), pp785-795.

Ellis, J.A. and Lepolesa, P.M., A Las Vegas Colouring Algorithm. *The Computer Journal*, **32**, 5, (1989), pp474-476.

Fleurent, C and Ferland, J.A., Genetic and hybrid algorithms for graph colouring. *Annals of Operations Research* **63**, (1996), pp437-461

Foxley, E. and Lockyer, K., The Construction of Examination Timetables by Computer. *Computer Journal*, **11**, (1968), pp264-268.

Gambardella, L.M. and Dorigo, M., Ant-Q: A Reinforcement Learning Approach to the Travelling Salesman Problem. Proceedings of ML-95, Twelfth International Conference on Machine Learning, Ed. Kaufmann, M., (1995), pp252-260.

Gambardella, L.M. and Dorigo, M., An Ant Colony System Hybridized with a New Local Search for the Sequential Ordering Problem, *INFORMS*, **12**, 3, (2000), pp237-255.

Gambardella, L.M., Taillard, E. and Agazzi, G., MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows. *New Ideas in Optimization*, Ed. Corne, D., Dorigio, M. and Glover, F., McGraw-Hill, (1999a), pp63-76.

Gambardella, L.M., Taillard, E. and Dorigo, M., Ant Colonies for the Quadratic Assignment Problem. *Journal of the Operational Research Society*, **50**, (1999b). pp.167-175.

Glover, F., Tabu Search - Part I, *ORSA Journal of Computing* **1**, No.3, (1989), pp190-206.
Goldberg, D.E, Genetic Algorithms in Search, Optimisation and Machine Learning. Addison-Wesley, Reading, Mass. (1989).

Holland, J., Adaptation in Natural and Artificial Systems, University of Michigan Press, (1975).

Jog, P., Suh, J.Y. and Gucht, D. Van, The effects of population size, heuristic crossover and local improvement on a genetic algorithm for the travelling salesman problem. Proceedings of the Third International Conference on Genetic Algorithms. Ed. Schaffer, J.D., (1989), pp110-115

Johnson, D., Timetabling University Examinations. *Journal of the Operatoinal Research Society*, **41**, 1, (1990), pp39-47.

Johnson, D.S., Aragon, C.R., McGeoch, L.A. and Schevon, C., Optimization by Simulated Annealing: An Experimental Evaluation; Part II, Graph Colouring and Number Partitioning. *Operational Research*, **39**, 3, May-June, (1991), pp378-406.

Julstrom, B.A., Comparing Darwinian, Baldwinian, and Lamarckian Search in a Genetic Algorithm for the 4-Cycle Problem. Technical Report. Department of Computer Science, St. Cloud State University, (2000).

Kempe, A.B., On the Geographical Problem of the Four Colours. *American Journal of Mathematics*, **2**, 3, (1879), pp193-200.

Kiaer, L. and Yellen, J., Weighted Graphs and University Course Timetabling. *Computers and Operations Research*, **19**, 1, (1992), pp59-67.

Kirkpatrick, S., Gelatt, C. and Vecchi, P., Optimisation by Simulated Annealing, *Science*, **220** (1983), pp671-679.

Kubale, M. and Kusz, E., Computational Experience with Implicit Enumeration Algorithms for Graph Colouring. In Proceedings of the WG'83 International Workshop on Graph Theoretic Concepts in Computer Science. Eds. Nagl, M. and Perl, J., Trauner Verlag, Linz (1983), pp167-176.

Laporte, G. and Desroches, S., Examination Timetabling By Computer. *Computers and Operations Research*, **11**, 4, (1984), pp351-360.

Leighton, F.T., A Graph Coloring Algorithm for Large Scheduling Problems. *Journal of Research*, **84**, 6, (1979), pp489-503.

Lofti, V. and Cerveny, R., A Final-Exam-Scheduling Package. *Journal of the Operational Research Society*, **42**, No.3., (1991), pp205-216.

Maniezzo, V. Exact and Approximate Nondeterministic Tree-Search Procedures for the Quadratic Assignment Problem. *INFORMS Journal on Computing*, **11**, 4, (1999), pp358-369.

Maniezzo, V. and Carbonaro, A., An ANTS Heuristic for the Frequency Assignment Problem. *Future Generation Computer Systems*, **16**, (2000), pp927-935

Maniezzo, V. and Colorni, A., The Ant System Applied to the Quadratic Assignment Problem. *IEEE Trans. Knowledge and Data Engineering*, **11**, 5, (1999), pp769-778.

Maniezzo, V., Colorni, A. and Dorigo, M., The Ant System Applied to the Quadratic Assignment Problem, IRIDIA Technical Report IRIDIA/94-28, Universite Libre de Bruxelles, Belgium. (1994)

Matula, D.W., Marble, G. and Isaacson, J.D., Graph Coloring Algorithms. Graph Theory and Computing,Ed. Read, R.C. Academic Press, New York and London, (1972), pp109-122.

Mehta, N.K., The Application of a Graph Coloring Method to an Examination Scheduling Problem. *INTERFACES*, **11**, 5, (1981), pp57-61.

Merlot, L.T.G., Boland, N., Hughes, B.D. and Stuckey, P.J., A Hybrid Algorithm for the Examination Timetabling Problem. In Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling. Eds Burke. E. and De Causmaecker. P, (2002), pp348-371.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. and Teller, E., Equation of state calculation by fast computing machines, *J. Chemical Physics*, **21**, (1953), pp1087-1091.

Michel, R. and Middendorf, M., An ACO Algorithm for the Shortest Common Supersequence Problem. *New Ideas in Optimization*, Ed. Corne, D., Dorigio, M. and Glover, F., McGraw-Hill, (1999), pp51-61.

Morgenstern, C., Algorithms for general graph colouring. Doctoral dissertation, Department of Computer Science, University of New Mexico, (1989a).

Morgenstern, C. and Shapiro, H., Chromatic number approximation using simulated annealing. Technical Report CS86-1, Department of Computer Science, University of Mexico, (1989b).

Moscato, P. and Norman, M. G., A memetic approach for the travelling salesman problem. In Proceedings of the International Conference on Parallel Computing and Transputer Applications, (1992), pp177-186.

Paechter, B., Luchain. H., Cumming, A. and Petriuc, M., Two Solutions to the General Timetable Problem Using Evolutionary Methods. In Proceedings of the IEEE Conference of Evolutionary Computation, (1994), pp305-310.

Patnaik, D. and Hosking, R.J., APL-based on-line examination scheduler. *APL Quote Quad* **15**, 3 (1985), pp21-26.

Papadimitriou, C. D. and Steiglitz, K., Combinatorial optimization: Algorithms and complexity. Prentice Hall, Englewood Cliffs, NJ. (1982).

Peck, J.E.I. and Williams, M.R., Algorithm 286 Examination Scheduling. *Communications of the ACM*, **9**, 6, (1966), pp433-434.

Peemöller, J., A Correction to Brelaz's Modification of Brown's Coloring Algorithm. *Communications of the ACM*, **26**, 8, (1983), pp595-597.

Petford, A.D. and Welsh, D.J.A., A Randomised 3-Colouring Algorithm. *Discrete Mathematics*, **74**, (1989), pp253-261.

Radcliffe, N.J. and Surry, P.D., Formal Memetic Algorithms. Evolutionary Computing. *Lecture Notes in Computer Science* **865**, Ed. Fogarty Terence, C. Springer-Verlag, (1994), pp1-16.

Reeves, R., Genetic Algorithms and Neighbourhood Search. Proceedings of the AISB Workshop. Lecture Notes in Computer Science **865**, Ed. Fogarty, T. C., Springer-Verlag, (1994), pp115-130.

Ross, P., Corne, D. and Fang, H-L., Improving Evolutionary Timetabling with Delta Evaluation and Directed Mutation. Parallel Problem Solving from Nature- PPSN III. *Lecture Notes in Computer Science* **866**, Eds. Davidor, Y., Schwefel, H-P. and Manner, R. Springer-Verlag, (1994), pp556-565.

Ross, P. and Corne, D., Comparing Genetic Algorithms, Simulated Annealing, and Stochastic Hillclimbling on Timetabling Problems. Evolutionary Computing. *Lecture Notes in Computer Science* **993**, Ed. Fogarty T.C. Springer-Verlag, (1995), pp94-102.

Ross, P., Hart, E. and Corne, D., Some Observations about GA-Based Exam Timetabling. Practice and Theory of Automated Timetabling II. Eds E.Burke and M.Carter, *Lecture Notes in Computer Science* **1408**, (1998),pp. 115-129

Schoonderwoerd, R., Holland, O., Bruten, J. and Rothkrantz, L., Ant-based Load Balancing in Telecommunications Networks. *Adaptive Behaviour*, **5**, 2, (1994), pp169-207.

Socha, K., Knowles, J. and Sampels, M., A MAX-MIN Ant System for the University Course Timetabling Problem, Proceedings of ANTS 2002, 3rd International Workshop on Ant Algorithms. *Lecture Notes in Computer Science* **2463**. Eds. Dorigo, M., Di Caro, G. and Sampels, M., (2002), pp1-13

Socha, K., Sampels, M. and Manfrin, M., Ant Algorithms for the University Course Timetabling Problem with regard to state of the art. Proceedings of the 3rd European Workshop on Evolutionary Computation in Combinatorial Optimization, EvoCOP, Essex, UK. *Lecture Notes in Computer Science* **2611**, (2003). pp334-345

Stützle, T., Parallelization Strategies for Ant Colony Optimization. Parallel Problem Solving from Nature - PPSN V. *Lecture Notes in Computer Science* **1498**, Eds. Eiben, A.E., Back, T., Schoenauer, M. and Schwefel, H-P. Springer-Verlag, (1998), pp722-731.

Stützle. T. and Hoos, H., Improvements on the Ant-System: Introducing the MAX-MIN Ant System. Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms, Eds. Smith, G.D., Steele, N.C. and Albrecht, R.F.Springer-Verlag, Vienna, (1996), pp245-249.

Stützle, T. and Hoos, H., MAX-MIN Ant System and Local Search for Combinatorial Optimisation Problems. 2nd International Conference on Metaheuristics - MIC97, Spohia-Antipolis, France, July, (1997).

Taillard, E., Benchmarks for basic scheduling problems. *European Journal of Operations Research* **64**, (1993), pp278-285

Taillard, E.D. and Gambardella, L.M., Adaptive Memories for the Quadratic Assignment Problem. IDSIA Technical Report, Lugano, Switzerland, (1997).

Thompson, J.M, Examination Scheduling Using Simulated Annealing. Doctoral dissertation, EBMS, University of Wales Swansea, (1995).

Thompson, J.M. and Dowsland, K.A., A Robust Simulated Annealing Based Examination Timetabling System. *Computers and Operational Research*, **25**, No.7/8, (1998), pp637-648.

Tillet, P.I., An operations research approach to the assignment of teachers to courses. *Socio-Econ. Plan. Sci.* **9**, (1975), pp101-104

Tripathy, A., A Lagrangean Relaxation Approach to Course Timetabling. *Journal of the Operational Research Society*, **31**, 7, (1980), pp599-603.

Varela, G. N. and Sinclair, M. C., Ant Colony Optimisation for Virtual-Wavelength-Path-Routing and Wavelength Allocation, Proceedings of the Congress on Evolutionary Computation, **3**, (1999), pp1809-1816

Vesel, A. and Zerovnik, J., How Well Can Ants Colour Graphs? *Journal of Computing and Information Technology - CIT* **8**, (2000), pp.131-136.

Watkins, C. J. C. H., Learning with Delayed rewards, Doctoral Dissertation, University of Cambridge, (1989).

Welsh, D.J.A. and Powell, M.B., An Upper Bound for the Chromatic Number of a Graph and its Application to Timetabling Problems. *Computer Journal*, **10**, (1967), pp85-86.

White, G.M. and Chan, P-W., Towards the Construction of Optimal Examination Schedules. *INFOR*, **17**, No.3, (1979), pp219-229.

White, G.M. and Haddad, M., An Heuristic Method for Optimizing Examination Schedules Which Have Day and Evening Courses. *Comput. Educ.*, **7**, 4, (1983), pp235-238.

White, G.M. and Xie, B.S., Examination Timetables and Tabu Search With Longer-Term Memory. Practice and Theory of Automated Timetabling III. *Lecture Notes In Computer Science* **2079**, Eds. Burke, E. and Erben, W. Springer-Verlag, (2000), pp85-103.

White, T., Kaegi, S. and Oda, T., Revisiting Elitism in Ant Colony Optimisation., *Lecture Notes in Computer Science* **2723**, Springer-Verlag Berlin Heidelberg, (2003), pp122-133

White, T. Pagurek, B. and Oppacher, F., ASGA: Improving the Ant System by Integration with Genetic Algorithms. Carleton University. Internal Report. (1997).

Wood, D.C., A System for Computing University Examination Timetables. *The Computer Journal* **11**, (1968), pp41-47

Wood, D.C, A Technique for Colouring a Graph Applicable to Large Scale Timetabling Problems. *The Computer Journal*, **12**, (1969), pp317-319.

Appendix 1.1

GLOSSARY OF TERMS

This section defines terms that will enable a clearer understanding of this work. This glossary has been divided into two main sections, timetabling terms and graph theoretic terms.

Timetabling Terms

A feasible timetable accommodates all the exams within the prespecified number of timeslots without requiring any student to sit at least two exams during the same sitting.

Two exams are said to be in conflict if they have common students. Two exams are clashing if they are in conflict and are currently scheduled for the same timeslot.

An exam is time-windowed if it cannot be scheduled in a subset of timeslots.

An exam is pre-assigned if it must be scheduled in a certain timeslot.

Graph Theoretic Terms

A graph G(V,E) consists of the set of vertices V and edges E. If an edge connects vertices v_1 and v_2 then v_1 is said to be adjacent to or a neighbour of v_2 and v_2 is said to be adjacent to or a neighbour of v_1.

The degree of a vertex is the number of vertices adjacent to it.

A clique is a set of vertices, which are all adjacent to one another. The maximum clique is the clique containing the most vertices in a given graph G.

A vertex colouring of a graph (graph colouring) is the allocation of colours to vertices such that no pair of adjacent vertices are placed within the same colour group.

The degree of saturation of a vertex v_l is equal to the number of colours used on the coloured neighbours of v_l.

If the computational effort increases exponentially as the size of the problem increases, the problem is said to be NP-hard.

A meta-heuristic is a search strategy, which guides the operation of a subordinate heuristic. The guiding principle of a meta-heuristic is usually derived from artificial intelligence studies (AI), mathematics, biological science or physics.

ANTCOL Terms

α represents the bias exponent of the trail factor.

β represents the bias exponent of the desirability factor.

ρ represents the level of evaporation.

τ symbolises the trail factor.

η symbolises the desirability (also known as visibility).

N_A refers to the number of ants used per cycle.

N_C refers to the number of cycles used per experimental run.

Appendix 1.2

The Travelling Salesman Problem (TSP)

The TSP is extensively studied in literature. The TSP is the problem of a salesman who wants to find, starting at his home city, the shortest possible route through a set of customer cities and to return to his hometown. More formally, it can be represented by a complete weighted graph G=(V,E), with V being the vertex set (representing the cities) and E the set of edges that connects the vertices. Each edge is assigned a weight d_{ij}, which is the length of the edge that connects vertices (representing the distance between the cities). The TSP is the problem of finding a minimal length Hamiltonian circuit of the graph, where a Hamiltonian circuit is a closed tour visiting each of the cities once.

Appendix 2.1

Definitions of RLF and DSatur

Section 1.4 discussed eight variants of the ANTCOL algorithm, which differed according to the construction heuristic, that were used. Two main construction heuristics formed the basis of these variants and will be defined here.

Recursive Largest First Degree (RLF, Leighton (1979))

The RLF heuristic constructs each colour group in turn. RLF is a dynamic procedure, which means that the choice of colour for a vertex at step k depends on the colour assignments for other vertices during the previous $k-1$ steps.

Let set D represent vertices belonging to the uncoloured subgraph.
Let c represent the colour group that is being filled.
Let v_i be the i^{th} vertex belonging to D.
Let m be the number of members in D.
Let $deg_D(v)$ represent the degree of a vertex v with respect to the set D.

Initialise $c=0$

1. Set $c=c+1$. Choose first member of colour c according to $max\{deg_D(v)\}$
2. Set $m=m-1$. If $m=0$ then Goto 5.
3. Order D in descending order according to $deg_D(v)$.
4. Consider each vertex v_i (*for* $i=1,...,m$) in turn. When $\sum_{e=1}^{n} \deg(v_i, v_e) = 0$ (where n=number of vertices coloured in c) is observed, v_i is inserted into c. Goto 2. If no insertion is possible. Goto 1
5. End routine as all vertices have been coloured.

Degree of Saturation (DSatur, Brélaz (1979))

DSatur fills 'active' colour groups concurrently.

Let A represent the set of vertices in the uncoloured subgraph.

Let q represent the number of colours used.

Let $c_{min}(v)$ represent the lowest feasible colour for vertex v.

Let m be the number of vertices in A.

Let $deg_{sat}(u)$ represent the degree of saturation of vertex u.

Let $deg_A(v)$ represent the degree of vertex v with respect to the set A.

1. Choose first vertex v according to $max\{deg_A(v)\}$ and allocate v colour 1.
2. Set $m=m-1$. If $m=0$ then Goto 4.
3. Allocate the vertex v that satisfies $max\{deg_{sat}(v)\}$ with colour c_{min}. If $deg_{sat}(v_1)=deg_{sat}(v_2)$, for two vertices v_1 and v_2, then differentiate according to deg_A. If equality still exists, then choose randomly. Update $deg_{sat}(u)$ and $c_{min}(u)$ $\forall$ neighbours (of v) $u \in A$. Neighbours u are dealt with in descending order according to $deg_{sat}(u)$. The above tiebreaker rules apply. If no $c_{min}(u)$ can be found such that $c_{min}(u) \leq q$ then set $q=q+1$ and allocate $c_{min}(u)=q$.

End of routine

Appendix 2.2

Genetic Algorithms

Genetic Algorithms (GA) is an evolutionary optimisation approach to non-linear models, which stochastically develops generations of coded solution (a classical genetic algorithm represents individuals by strings) populations using a fitness statistic.

Holland (1975) is acclaimed as the originator of GA in its current form. GA is a probabilistic search approach, based on the ideas of evolutionary processes. The GA analogy is based on the Darwinian principle of survival of the fittest. An initial population is created containing a predefined number of solutions (chromosomes), each represented by a genetic string. New populations are achieved through the combination of genes from different population members (crossover) or by altering existing members of the population (mutation). Each solution has a fitness value.

It is conceptualised that the fittest members of a population will produce fitter offspring.

GA's consist of four main stages: Evaluation, Selection, Crossover and Mutation.

Evaluation – Measure the fitness of individual solutions and assigns them fitness scores.
Selection – Pseudo-randomly selects individuals of current population for development of next solution, bias according to fitness level.
Crossover – Takes two selected individuals of current population and combines about a crossover point thereby creating two individuals. Asexual reproduction is also possible.
Mutation – Randomly modifies the genes of an individual subject to a small mutation factor.

The algorithm terminates when a predetermined criteria has been satisfied e.g. time, set number of populations.

Appendix 3.1

Efficiency of construction heuristics

In this section we evaluate the potency of the construction heuristics that provide the basis of solution construction. The heuristics are regarded under two conditions – with and without pheromone. The former environment has parameter settings of $\alpha=2$, $\beta=1$ and $\rho=0.5$. For each construction heuristic, the average number of colours required to colour the graph is used as the benchmark statistic and is ranked in a range of 1 (best) to 8 (worst).

	HEC		EAR		TRENT			
Construction Heuristic	*Single Pass*	*With Trail*	*Single Pass*	*With Trail*	*Single Pass*	*With Trail*	*Mean Single Pass Rank*	*Mean With Trail Rank*
A	5	2	5	3	5	3	5	2.67
B	8	5	8	6	7	6	7.67	5.67
C	6	1	6	1	6	2	6	1.33
D	1	4	1	4	1	5	1	4.33
E	3	8	3	8	3	7	3	7.67
F	2	6	2	5	2	4	2	5
G	4	3	4	2	4	1	4	2
H	7	7	7	7	8	8	7.33	7.33

Figure A3.1 – Rank statistics for eight construction used with and without trail across data sets

The mean ranks indicate that the best construction heuristic to use with trail is construction heuristic C.

Appendix 3.2

The following sequence of Tables present the average and minimum number of colours required to colour the graphs representing HEC, EAR and TRENT respectively for combinations of the bias parameters α and β across the eight construction heuristics.

		Av					Best				
	α/β	*0*	*1*	*2*	*3*	*4*	*0*	*1*	*2*	*3*	*4*
A	*0*	22.60	21.74	20.95	20.35	19.91	19	18	18	17	17
	0.5	22.19	21.21	20.40	19.83	19.19	19	18	18	17	17
	1	21.44	20.42	19.67	19.24	18.99	18	17	17	17	17
	2	18.59	18.14	17.84	17.72	17.98	17	17	17	17	17
	3	18.18	18.18	18.13	18.29	18.08	17	17	17	17	17
	4	18.01	18.10	18.27	18.13	18.18	17	17	17	17	17
B	*0*	22.60	22.63	22.62	22.58	22.55	19	19	19	19	19
	0.5	22.19	22.29	22.32	22.32	22.31	19	19	19	19	19
	1	21.44	21.72	21.87	21.94	21.99	18	18	18	19	18
	2	18.59	19.25	19.80	20.38	20.80	17	17	17	17	18
	3	18.18	18.81	19.47	19.79	20.04	17	17	17	17	18
	4	18.01	19.13	18.99	19.53	18.89	17	17	17	17	17
C	*0*	22.60	22.02	21.44	20.93	20.53	19	19	18	18	18
	0.5	22.19	21.22	20.52	20.34	19.97	19	18	18	18	17
	1	21.44	20.66	19.94	19.47	19.15	18	18	17	17	17
	2	18.59	18.09	18.22	18.34	18.14	17	17	17	17	17
	3	18.18	18.19	18.28	18.08	18.11	17	17	17	17	17
	4	18.01	18.10	18.11	18.27	17.92	17	17	17	17	17
D	*0*	20.80	20.31	19.92	19.62	19.41	18	18	17	17	17
	0.5	20.68	20.13	19.74	19.48	19.31	18	17	17	17	17
	1	20.44	20.18	19.67	19.71	19.69	18	18	18	17	17
	2	20.97	19.06	19.24	19.79	19.60	18	18	17	17	17
	3	20.61	19.81	19.23	18.63	19.40	18	18	18	18	18
	4	20.41	19.80	19.60	19.59	19.02	18	18	18	18	18
E	*0*	20.80	20.87	20.91	20.97	21.01	18	18	18	18	18
	0.5	20.68	20.81	20.93	21.04	21.10	18	18	18	18	18
	1	20.44	20.77	21.09	21.13	21.20	18	18	18	18	18
	2	20.97	21.75	21.58	20.82	20.99	18	18	18	19	18
	3	20.61	21.58	21.98	21.59	20.62	18	18	18	19	19
	4	20.41	21.59	21.21	21.20	20.82	18	19	19	19	17
F	*0*	20.80	20.45	20.12	19.83	19.62	18	18	17	18	17
	0.5	20.68	20.24	19.87	19.61	19.42	18	18	18	17	17
	1	20.44	20.07	20.02	19.60	19.06	18	17	18	17	17
	2	20.97	20.02	18.86	19.41	19.23	18	18	18	18	17
	3	20.61	19.06	19.99	18.83	19.20	18	18	18	18	17
	4	20.41	19.62	19.22	18.82	19.60	18	18	18	18	18
G	*0*	21.72	21.88	21.09	20.82	20.58	18	18	18	18	18
	0.5	21.27	20.93	20.63	20.39	20.21	18	18	18	18	17
	1	20.55	20.25	19.98	19.77	19.69	18	17	17	17	17
	2	18.42	18.21	18.53	18.68	18.15	18	17	17	17	17
	3	18.42	18.38	18.36	18.20	18.36	18	17	17	17	17
	4	18.32	18.17	18.50	18.50	18.65	18	17	17	17	17
H	*0*	22.75	22.40	22.05	21.68	21.35	19	19	18	18	18
	0.5	22.51	22.14	21.76	21.39	21.05	19	19	18	18	17
	1	22.22	21.79	21.39	20.99	20.65	19	18	18	18	18
	2	21.66	21.17	20.74	20.34	20.02	18	17	17	17	17
	3	21.40	20.92	20.51	20.16	19.85	18	18	17	17	17
	4	21.27	20.80	20.43	20.07	19.75	18	18	17	17	17

Figure A3.2 – Average and minimum colours for range of α and β for HEC

		Av					Best				
	α/β	*0*	*1*	*2*	*3*	*4*	*0*	*1*	*2*	*3*	*4*
A	*0*	33.57	32.19	30.84	29.69	28.71	28	27	25	25	25
	0.5	33.01	31.41	29.90	28.62	27.66	29	26	25	24	24
	1	31.68	29.81	28.13	26.87	26.06	26	25	23	23	23
	2	25.94	25.27	24.16	23.84	23.59	23	23	22	23	23
	3	25.75	25.34	24.28	23.88	24.24	23	23	22	23	23
	4	25.61	25.00	24.88	23.82	24.00	23	23	23	22	23
B	*0*	33.57	33.61	33.60	33.56	33.52	28	28	29	28	28
	0.5	33.01	33.12	33.15	33.16	33.15	29	28	28	28	28
	1	31.68	32.03	32.27	32.46	32.57	26	28	28	28	28
	2	25.94	27.22	28.37	29.57	30.19	23	24	24	25	26
	3	25.75	26.24	27.02	28.03	28.78	23	24	24	25	24
	4	25.61	26.20	27.44	27.45	27.92	23	23	24	24	23
C	*0*	33.57	32.47	31.33	30.29	29.40	28	28	27	26	25
	0.5	33.01	31.71	30.36	29.16	28.22	29	26	25	25	24
	1	31.68	30.13	28.47	27.13	26.36	26	25	24	23	23
	2	25.94	24.81	24.20	23.47	23.96	23	23	22	22	22
	3	25.75	24.19	24.15	23.67	23.81	23	23	23	23	23
	4	25.61	24.97	24.21	23.78	24.18	23	23	23	23	23
D	*0*	29.40	28.29	27.50	26.46	25.81	26	25	24	24	23
	0.5	28.93	27.58	26.42	25.48	24.86	26	25	23	23	23
	1	27.75	26.12	24.16	23.70	23.40	25	24	23	23	23
	2	26.53	25.50	24.47	23.69	23.46	25	24	23	23	23
	3	27.43	25.28	24.27	23.84	23.83	26	24	23	23	23
	4	27.42	25.08	24.65	23.84	23.83	25	24	23	23	23
E	*0*	29.40	29.20	29.01	28.87	28.77	26	26	26	26	26
	0.5	28.93	28.74	28.61	28.52	28.46	26	25	26	26	26
	1	27.75	27.84	28.01	25.70	25.40	25	25	25	23	23
	2	26.53	25.50	24.47	23.69	23.46	25	24	23	23	23
	3	27.43	25.28	24.27	23.84	23.83	26	24	23	23	23
	4	27.42	25.08	24.65	23.84	23.83	25	24	23	23	23
F	*0*	29.40	28.61	27.79	27.05	26.45	26	25	25	24	23
	0.5	28.93	27.91	26.91	26.04	25.44	26	25	24	23	23
	1	27.75	26.31	24.44	24.03	24.30	25	24	23	23	23
	2	26.53	25.51	24.15	23.73	24.04	25	24	23	23	23
	3	27.43	25.69	24.67	23.86	24.04	26	24	23	23	23
	4	27.42	26.24	24.66	24.05	24.04	25	24	23	23	23
G	*0*	31.56	30.94	30.38	29.85	29.36	27	27	26	26	25
	0.5	30.48	29.80	29.23	28.71	28.28	27	25	25	25	24
	1	28.68	27.96	27.47	27.08	26.76	25	24	24	24	24
	2	25.00	24.88	24.67	24.52	24.54	23	23	23	23	23
	3	24.76	24.46	24.41	24.39	24.17	23	23	23	23	23
	4	24.90	24.13	24.30	24.05	24.25	23	23	23	23	23
H	*0*	34.10	33.37	32.61	31.88	31.19	29	28	28	27	26
	0.5	33.78	33.02	32.20	31.45	30.73	29	28	28	27	26
	1	33.37	32.52	31.66	30.82	30.07	28	28	27	26	25
	2	32.14	31.12	30.26	29.47	28.79	27	26	26	25	24
	3	31.52	30.61	29.83	29.09	28.42	26	26	25	24	24
	4	31.33	30.48	29.66	28.93	28.26	26	25	25	24	24

Figure A3.3 – Average and minimum colours for range of α and β for EAR

		Av					Best				
	α/β	*0*	*1*	*2*	*3*	*4*	*0*	*1*	*2*	*3*	*4*
A	*0*	29.95	28.34	26.93	25.83	25.01	26	24	23	22	22
	0.5	29.72	27.94	26.41	25.24	24.45	25	24	23	22	21
	1	29.13	27.24	25.56	24.37	23.65	25	24	22	21	21
	2	23.54	22.62	22.12	22.08	21.90	20	20	20	20	20
	3	22.65	22.49	21.74	21.97	21.59	20	21	20	20	20
	4	22.59	22.49	21.82	22.00	21.44	20	21	20	20	20
B	*0*	29.95	29.92	29.88	29.85	29.81	26	26	26	26	26
	0.5	29.72	29.71	29.69	29.66	29.66	25	25	25	26	26
	1	29.13	29.18	29.22	29.27	29.31	25	25	25	25	25
	2	23.54	24.15	24.95	26.31	26.92	20	21	21	22	22
	3	22.65	22.81	24.37	23.34	24.28	20	20	22	20	21
	4	22.59	23.28	23.80	23.81	24.55	20	21	21	21	20
C	*0*	29.95	28.71	27.53	26.56	25.84	26	24	23	23	22
	0.5	29.72	28.34	27.03	25.99	25.25	25	24	23	23	21
	1	29.13	27.65	26.18	25.07	24.42	25	24	22	22	21
	2	23.54	22.53	22.29	21.67	21.64	20	20	20	20	20
	3	22.65	22.31	21.86	21.36	21.68	20	20	20	20	20
	4	22.59	22.14	22.12	21.39	21.87	20	20	20	20	20
D	*0*	27.51	26.16	25.02	24.17	23.55	24	23	21	21	21
	0.5	27.34	25.74	24.41	23.52	22.95	24	23	21	21	21
	1	26.71	24.40	23.03	22.26	21.72	23	21	21	20	20
	2	24.59	23.70	22.87	22.44	21.86	23	22	21	21	21
	3	25.08	23.30	21.91	22.40	21.29	23	22	21	20	20
	4	25.26	23.48	22.66	21.66	21.64	24	22	21	20	21
E	*0*	27.51	27.41	27.34	27.28	27.25	24	24	24	24	24
	0.5	27.34	27.26	27.21	27.16	27.13	24	24	24	24	24
	1	26.71	26.64	26.63	26.66	26.67	23	23	23	24	23
	2	24.59	24.76	24.93	25.47	25.14	23	23	23	23	23
	3	25.08	24.91	25.27	25.29	25.27	23	23	23	24	24
	4	25.26	25.45	24.71	25.26	25.03	24	24	23	24	23
F	*0*	27.51	26.50	25.59	24.87	24.36	24	23	22	22	22
	0.5	27.34	26.13	25.05	24.25	23.73	24	23	22	21	21
	1	26.71	24.97	23.47	22.60	22.03	23	21	21	20	20
	2	24.59	23.19	22.52	21.93	21.16	23	22	21	21	20
	3	25.08	23.73	23.06	22.26	21.66	23	22	21	21	20
	4	25.26	23.84	22.87	22.26	22.05	24	22	21	20	21
G	*0*	28.47	27.84	27.25	26.71	26.25	25	24	23	23	22
	0.5	27.89	27.16	26.54	26.00	25.59	24	23	23	22	22
	1	26.78	25.97	25.35	24.85	24.50	23	22	22	22	21
	2	22.11	22.03	21.52	22.13	21.45	20	20	20	20	20
	3	21.82	21.65	21.86	22.39	21.41	20	20	21	21	20
	4	22.32	21.70	22.07	21.94	21.86	21	20	20	20	20
H	*0*	31.35	30.57	29.76	29.00	29.00	27	26	25	25	24
	0.5	31.19	30.30	29.55	28.76	28.04	27	27	25	24	24
	1	31.01	30.16	29.29	28.45	27.69	27	26	25	24	23
	2	30.44	29.39	28.32	27.42	26.66	26	25	24	23	23
	3	29.78	28.64	27.71	26.88	26.18	25	24	23	22	22
	4	29.54	28.38	27.49	26.69	25.99	25	24	23	22	22

Figure A3.4 – Average and minimum colours for range of α and β for TRENT

Appendix 3.3

Performance across α and β

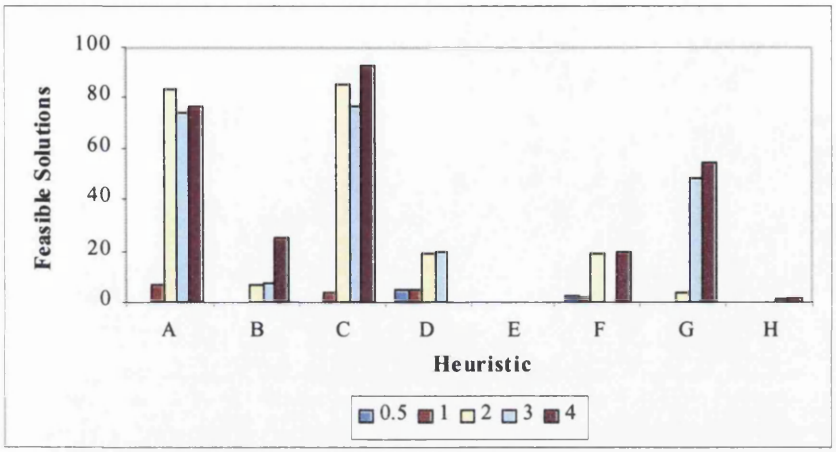


Figure A3.1 – Proportion of Feasible Solutions across α when β = 2 for HEC

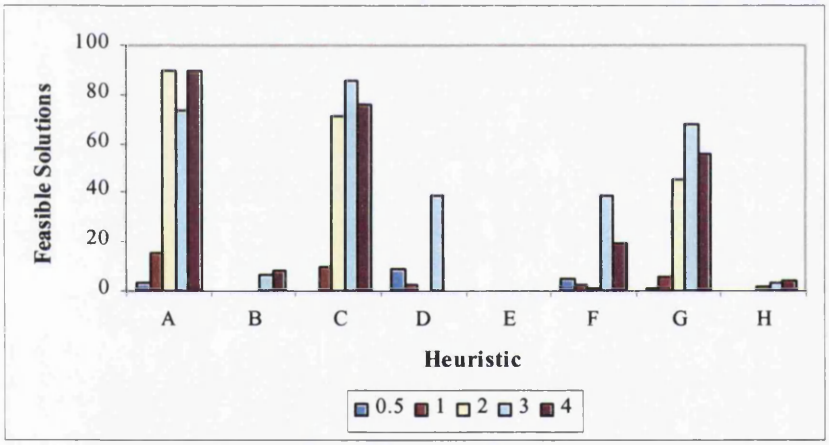


Figure A3.2 – Proportion of Feasible Solutions across α when β = 3 for HEC

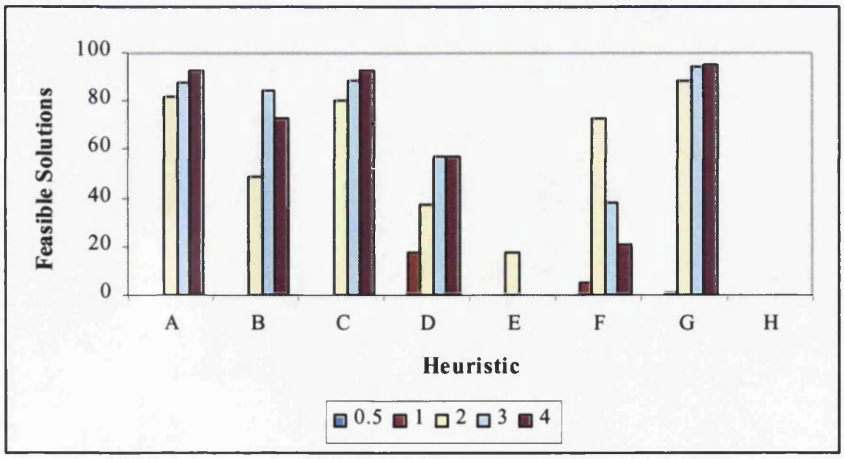


Figure A3.3 – Proportion of Feasible Solutions across α when β = 4 for HEC

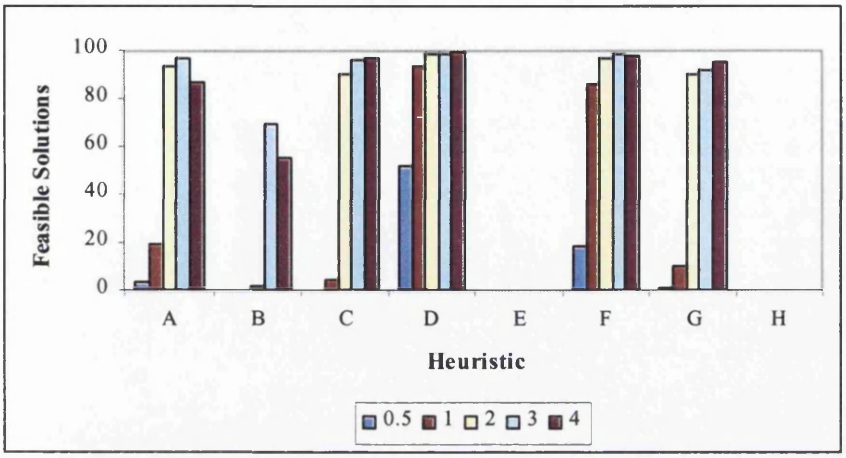


Figure A3.4 – Proportion of Feasible Solutions across α when $\beta = 2$ for EAR

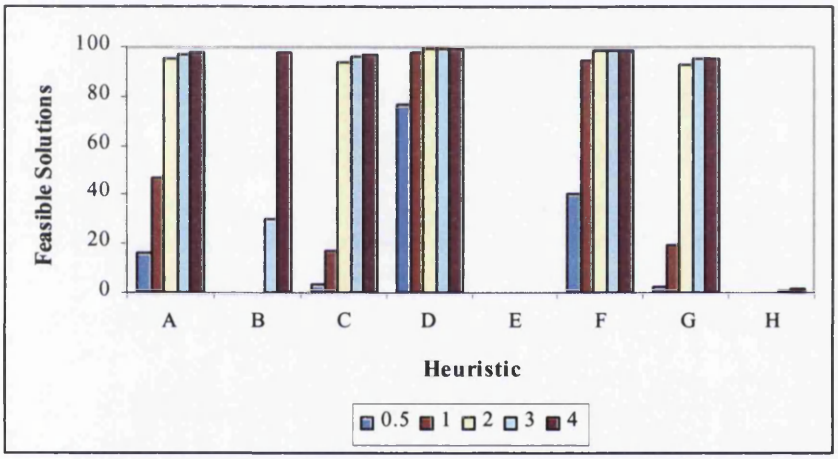


Figure A3.5 – Proportion of Feasible Solutions across α when $\beta = 3$ for EAR

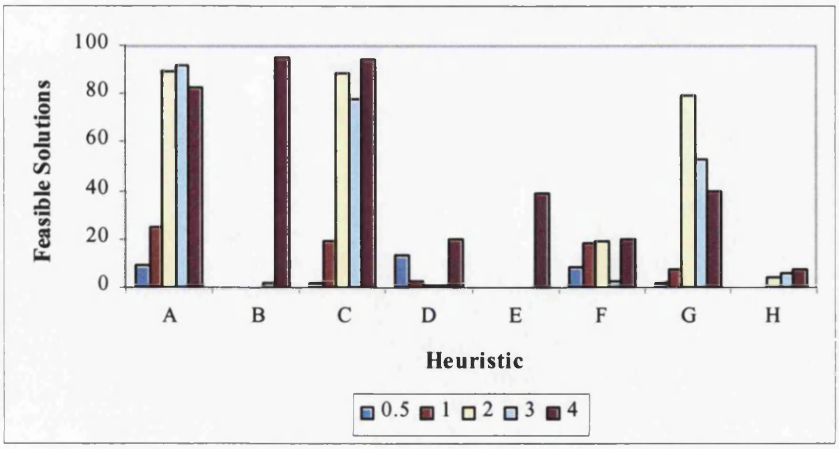


Figure A3.6 – Proportion of Feasible Solutions across α when $\beta = 4$ for EAR

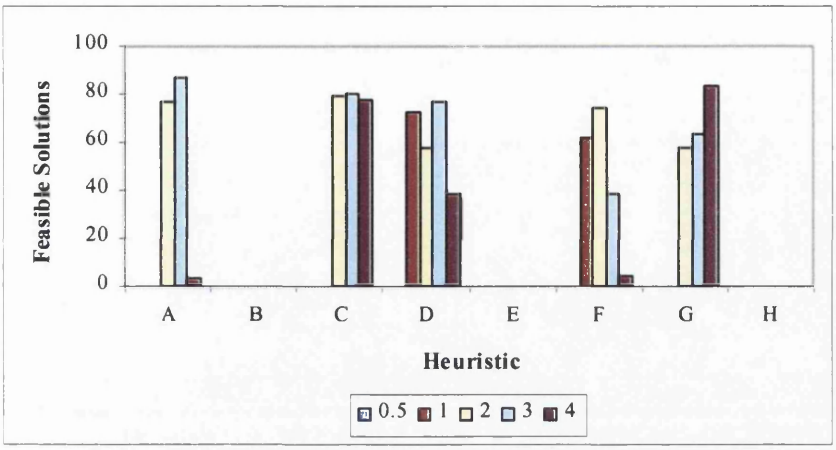


Figure A3.7 – Proportion of Feasible Solutions across α when $\beta = 2$ for TRENT

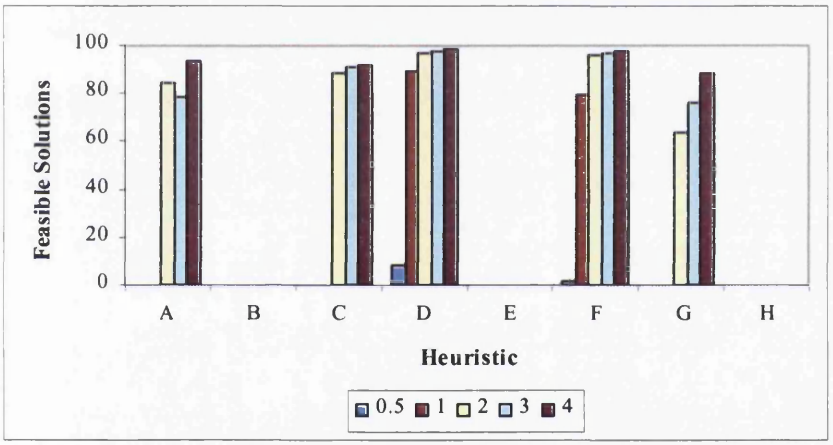


Figure A3.8 – Proportion of Feasible Solutions across α when $\beta = 3$ for TRENT

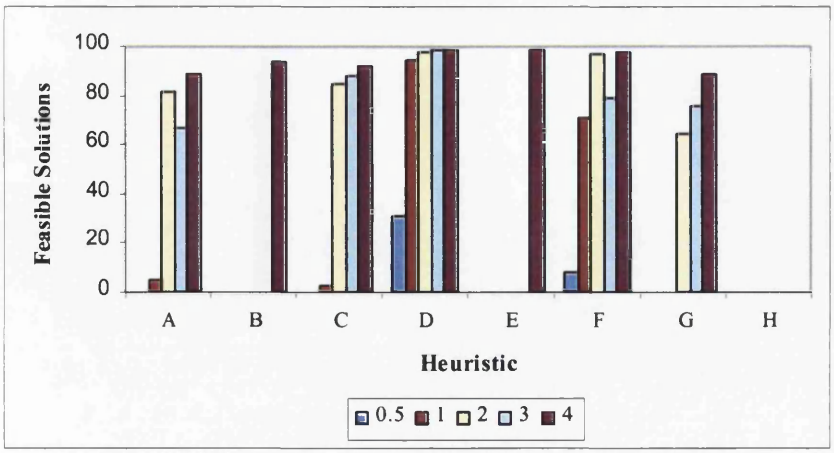


Figure A3.9 – Proportion of Feasible Solutions across α when $\beta = 4$ for TRENT

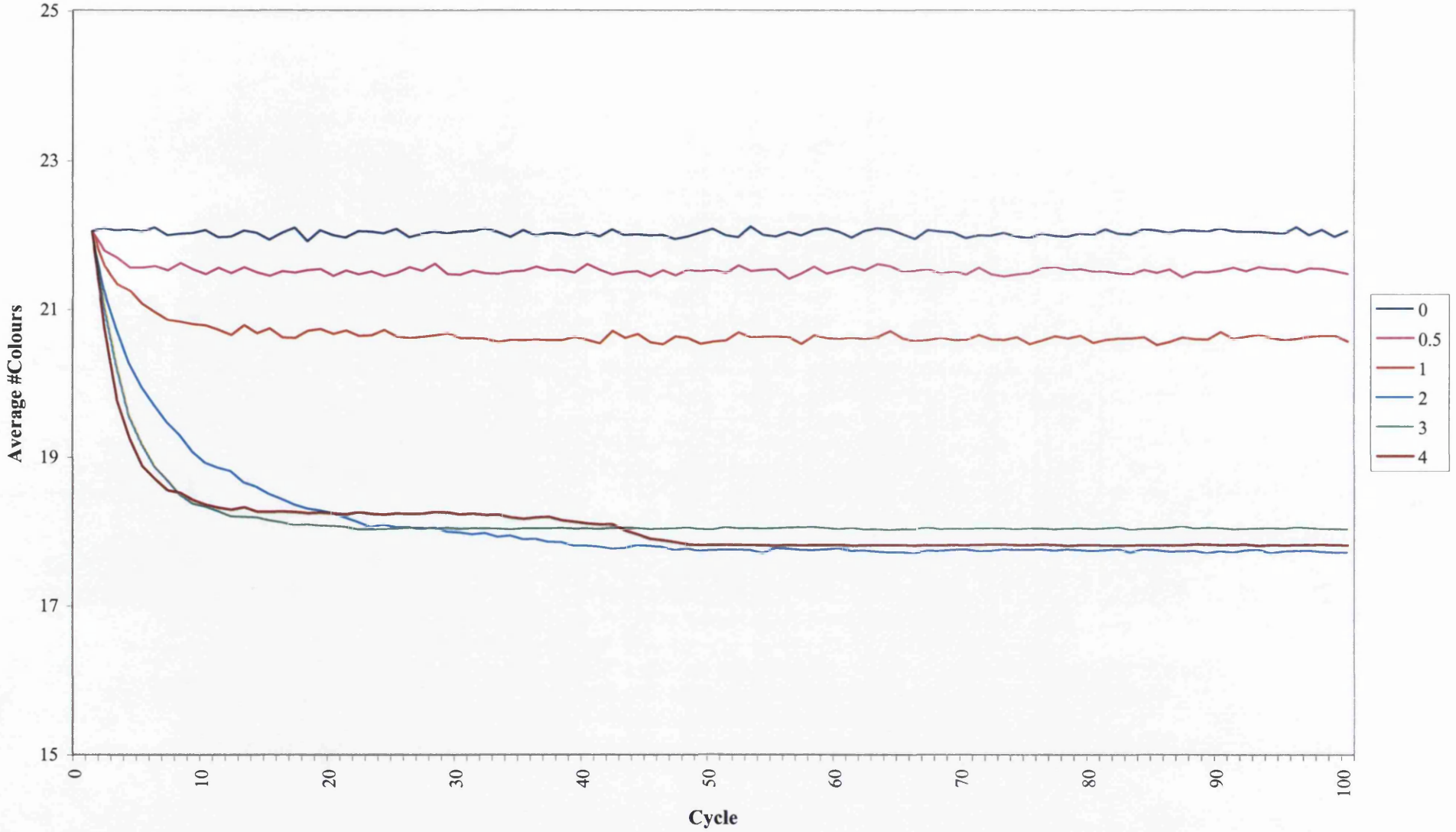


Figure A3.10 - Influence of alpha for HEC

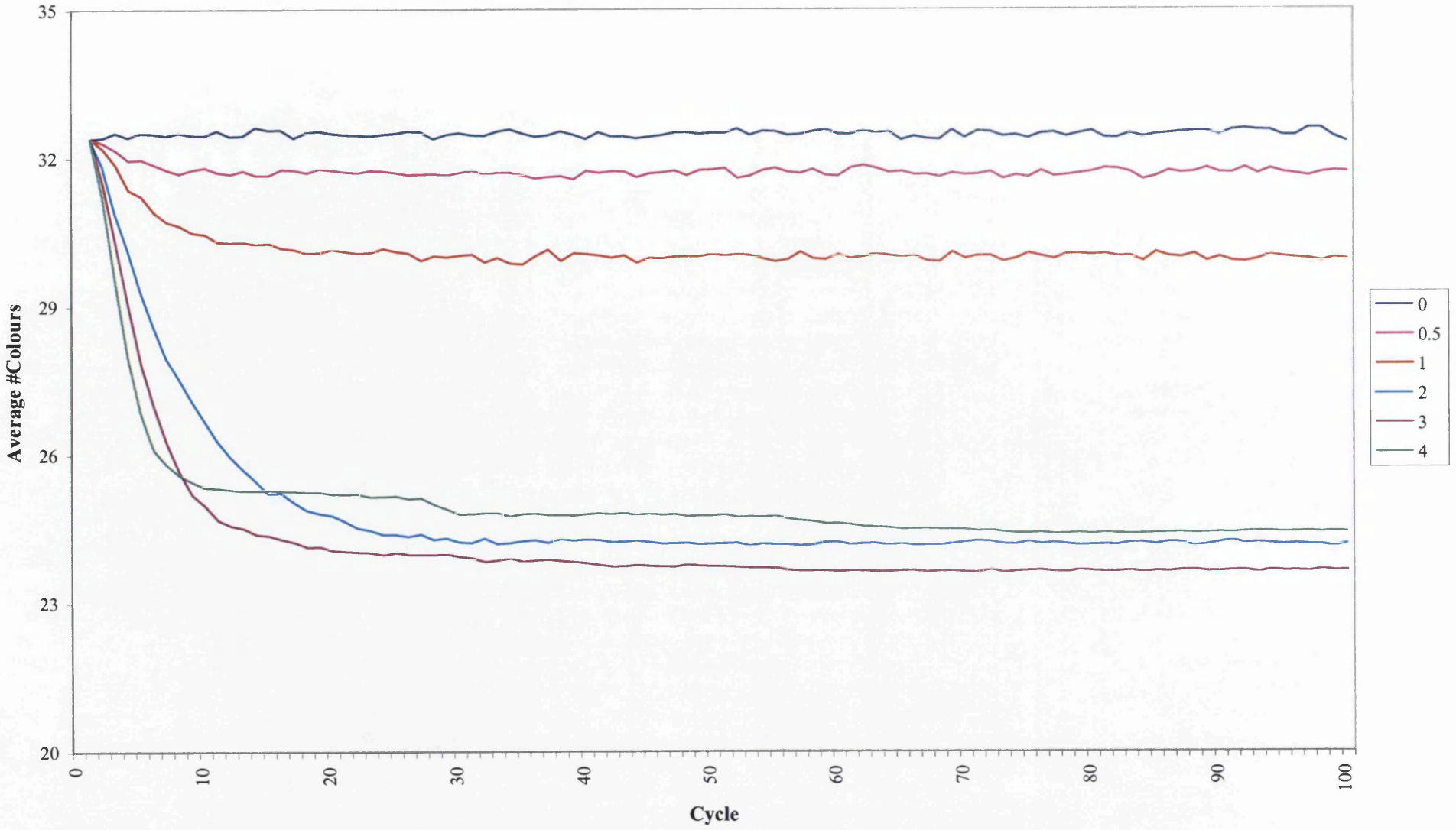


Figure A3.11 - Influence of alpha for EAR

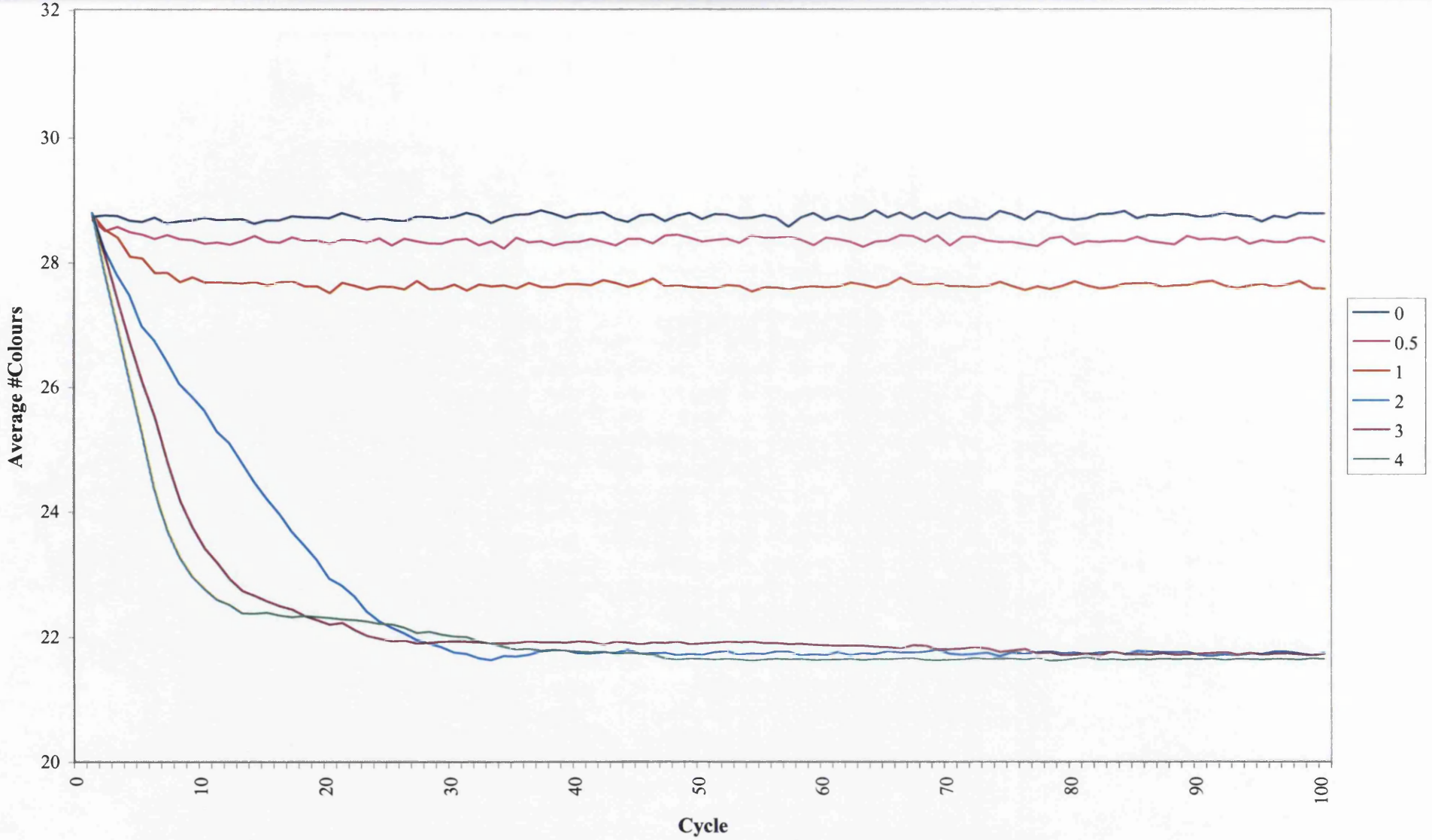


Figure A3.12 - Influence of alpha for TRENT

Appendix 3.4

Sensitivity of ρ

		ρ						
Heuristic		*0.01*	*0.10*	*0.30*	*0.50*	*0.70*	*0.90*	*0.99*
A	*Best*	17	17	17	17	17	17	17
	Av	18.23	18.24	18.14	18.14	18.17	18.73	19.26
B	*Best*	17	17	17	17	17	17	18
	Av	18.75	18.80	19.29	18.91	19.25	19.89	20.67
C	*Best*	17	17	17	17	17	17	17
	Av	17.77	18.08	18.30	18.05	18.45	18.65	19.37
D	*Best*	18	18	18	18	17	18	18
	Av	19.43	19.42	19.64	19.07	19.27	19.85	19.61
E	*Best*	19	19	19	19	18	18	18
	Av	21.58	21.95	21.56	21.00	21.92	22.01	21.64
F	*Best*	18	18	18	18	18	18	18
	Av	19.81	19.43	19.83	19.62	19.83	19.63	19.23
G	*Best*	17	17	17	17	17	17	17
	Av	18.07	18.16	18.21	18.21	18.97	18.90	19.92
H	*Best*	18	18	18	17	18	18	18
	Av	21.17	21.17	21.16	21.17	21.17	21.22	21.34

Table A3.5 – Average and Best colours for range of ρ for HEC

		ρ						
Heuristic		*0.01*	*0.10*	*0.30*	*0.50*	*0.70*	*0.90*	*0.99*
A	*Best*	23	23	23	23	23	23	23
	Av	25.09	25.09	24.82	25.05	25.27	26.20	27.82
B	*Best*	24	24	24	24	24	24	25
	Av	26.89	27.11	27.18	27.12	27.47	28.33	29.99
C	*Best*	23	23	23	23	23	22	23
	Av	24.60	24.96	25.13	25.23	25.37	26.07	27.88
D	*Best*	24	24	24	24	24	24	24
	Av	25.47	25.10	24.91	25.32	25.49	25.73	25.69
E	*Best*	25	25	25	25	25	25	25
	Av	25.74	26.47	26.87	27.07	26.42	27.08	27.27
F	*Best*	24	24	24	24	24	24	24
	Av	25.48	26.05	25.12	25.52	25.74	25.84	26.18
G	*Best*	23	23	23	23	23	24	24
	Av	24.29	24.28	24.67	24.85	24.93	26.45	28.42
H	*Best*	26	26	26	26	25	26	27
	Av	31.10	31.10	31.11	31.12	31.17	31.44	31.84

Table A3.6 – Average and Best colours for range of ρ for EAR

Heuristic		ρ						
		0.01	*0.10*	*0.30*	*0.50*	*0.70*	*0.90*	*0.99*
A	*Best*	20	20	20	20	20	20	22
	Av	21.90	21.65	22.00	22.45	22.79	24.01	25.98
B	*Best*	21	21	21	20	20	21	24
	Av	23.79	23.69	23.92	23.94	24.59	26.60	28.08
C	*Best*	20	20	20	20	20	20	22
	Av	22.29	22.38	22.19	22.29	22.98	24.39	26.42
D	*Best*	22	22	22	22	22	21	21
	Av	23.47	23.11	23.11	23.52	23.38	23.34	23.79
E	*Best*	23	23	23	23	23	23	23
	Av	24.54	24.72	24.76	24.97	24.82	24.94	25.47
F	*Best*	22	22	22	22	22	22	22
	Av	23.31	23.15	23.70	23.54	23.75	23.64	24.15
G	*Best*	20	20	20	20	20	20	20
	Av	21.47	21.64	21.37	22.03	22.05	24.47	27.05
H	*Best*	25	25	25	25	25	25	25
	Av	29.39	29.39	29.37	29.39	29.42	29.59	29.83

Table A3.7 – Average and Best colours for range of ρ for TRENT

Appendix 3.5

Sensitivity of N_A

Heuristic		N_A = 1	5	10	25	50	75	100	150	200
A	*Best*	18	17	17	17	17	17	17	17	17
	Av	19.09	18.30	18.64	18.15	18.34	17.97	18.22	18.31	18.19
B	*Best*	18	18	18	17	17	17	17	17	17
	Av	20.29	19.60	19.36	19.14	19.48	19.29	19.03	18.97	19.33
C	*Best*	18	18	17	17	17	17	17	17	17
	Av	18.91	18.59	18.29	18.13	18.38	17.99	18.23	18.01	18.13
D	*Best*	19	18	18	18	18	17	18	18	18
	Av	19.87	19.64	19.83	20.00	19.06	19.63	18.88	19.25	19.81
E	*Best*	19	19	19	19	18	18	19	18	18
	Av	20.85	20.81	21.00	21.21	21.37	21.96	21.00	21.56	21.40
F	*Best*	18	18	18	18	18	17	18	18	18
	Av	20.01	19.05	20.60	20.00	19.25	20.15	19.62	19.44	19.26
G	*Best*	17	17	17	17	17	17	17	17	17
	Av	18.51	18.74	18.52	18.45	18.37	18.31	18.21	18.54	18.72
H	*Best*	20	19	18	18	18	18	17	18	18
	Av	24.55	21.67	21.33	21.24	21.19	21.17	21.17	21.14	21.14

Table A3.8 – Average and Best colours for range of ants per cycle for HEC

Heuristic		N_A = 1	5	10	25	50	75	100	150	200
A	*Best*	25	24	23	23	23	23	23	23	23
	Av	27.92	26.14	25.92	25.61	25.40	25.47	24.93	25.30	25.17
B	*Best*	26	25	24	24	24	24	24	24	23
	Av	30.13	28.50	27.57	27.49	27.44	27.31	27.37	27.31	26.99
C	*Best*	25	23	23	23	23	23	23	22	22
	Av	27.93	26.55	25.79	25.01	25.16	24.85	25.27	24.67	24.82
D	*Best*	25	24	24	24	24	24	24	24	24
	Av	27.00	25.76	25.40	24.97	25.14	25.13	25.31	25.65	24.94
E	*Best*	26	26	26	25	25	25	25	25	25
	Av	28.00	27.35	27.53	26.92	26.73	27.44	27.06	27.07	26.35
F	*Best*	26	25	24	24	24	24	24	24	24
	Av	27.48	26.33	25.60	25.91	25.19	25.17	25.17	24.95	25.16
G	*Best*	23	23	23	23	23	23	23	23	23
	Av	25.67	24.79	24.64	24.72	24.61	24.37	24.85	24.82	24.73
H	*Best*	31	28	27	26	27	27	26	26	25
	Av	41.27	33.53	31.86	31.26	31.17	31.13	31.12	31.09	31.07

Table A3.9 – Average and Best colours for range of ants per cycle for EAR

Heuristic		N_A								
		1	*5*	*10*	*25*	*50*	*75*	*100*	*150*	*200*
A	*Best*	22	21	20	21	20	20	20	20	20
	Av	25.11	23.39	22.90	22.54	22.57	22.47	22.63	22.26	22.35
B	*Best*	23	22	21	20	21	21	21	21	21
	Av	27.19	26.10	25.17	24.57	24.18	24.25	24.43	24.34	24.11
C	*Best*	22	21	20	20	20	20	20	20	20
	Av	25.25	23.52	23.26	22.36	23.01	22.52	22.67	22.49	22.68
D	*Best*	24	22	22	21	22	22	21	21	22
	Av	25.47	23.61	23.61	23.19	23.01	23.33	23.17	23.52	23.34
E	*Best*	24	24	23	23	23	23	23	23	23
	Av	26.27	25.58	25.13	25.17	24.80	24.78	25.31	24.96	25.47
F	*Best*	24	22	21	22	22	21	22	22	22
	Av	25.77	24.03	22.94	23.12	23.77	22.67	23.21	23.57	23.36
G	*Best*	20	21	20	20	20	20	20	20	20
	Av	23.61	22.98	22.11	22.11	21.84	21.93	22.03	21.92	21.73
H	*Best*	28	28	26	26	25	25	25	25	25
	Av	37.78	32.41	30.28	29.55	29.42	29.40	29.39	29.56	29.35

Table A3.10 – Average and Best colours for range of ants per cycle for TRENT

Appendix 4.1

Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling. Eds Burke. K. and De Causmaecker. P., (2002), pp397-399.

Examination Timetabling with Ants.

Kathryn A. Dowsland[1], Nicholas Pugh[2] and Jonathan Thompson[3]

[1]Gower Optimal Algorithms Limited and University of Nottingham, [2]University of Wales Swansea, [3]University of Wales, Cardiff. Email: ThompsonJM1@cardiff.ac.uk

Introduction

This paper is concerned with the potential of ant algorithms as a useful tool for the solution of examination scheduling problems. Ant algorithms are a family of evolutionary meta-heuristics loosely based on the way that ants cooperate by laying pheromone trails. Initially applications focussed on routing and related problems where the analogy between the biological metaphor and the optimisation problem is a natural one. However, there is a growing interest in applications in other areas. The decision to experiment with ant algorithms for examination scheduling was motivated by the following observations.

One of the basic requirements of the approach is a greedy construction heuristic that can be randomised so that the next option is chosen with a probability proportional to its perceived quality (referred to as the visibility). These probabilities are changed as the algorithm progresses using a suitably defined 'trail' that encapsulates the important features of the solution. In the first cycle a population of solutions are generated using the randomised greedy approach. These are then used to update the trail, such that elements of higher quality solutions receive more trail, and those that appear in poorer solutions or not at all receive less trail. The trail is then used to adjust the probabilities for the randomised greedy construction in future cycles so that the probability of selecting elements of previous good solutions is increased. Thus an ant algorithm can be seen as a way of fine-tuning the probabilities in a greedy construction approach. This is encouraging as there has been considerable success in the solution of examination scheduling problems based on greedy construction heuristics. Although some of the earliest approaches using greedy construction alone cannot compete with today's state of the art approaches, many more recent implementations combine a greedy approach with other heuristics to provide

improved performance. For example, Ross et al. [4] use a GA in combination with a greedy or randomised greedy construction, while Carter et al. [1] describes a system based on greedy construction with some backtracking.

Further encouragement is provided by Costa and Hertz [2]. It is well known that the underlying model for examination scheduling is that of graph colouring. In [2] it is suggested that ant algorithms can be used for a variety of allocation type problems and the graph colouring problem is used to illustrate a family of algorithms they refer to as ANTCOL. Other successes in related areas include Maniezzo and Carbonaro's [3] work on the frequency assignment problem. However, it should be noted that Vesel and Zerovnik [6] report the results of a set of experiments that disagree with the conclusions of [2].

The scope of the investigation.

Although the graph colouring problem can model most of the constraints found in a typical instance of the examination-scheduling problem, it cannot incorporate secondary objectives such as second order conflict. Any successful approach to the problem must obviously be able to deal with such issues. However, in the work described here we restrict ourselves to finding feasible timetables, as defined by those features that can be included within the graph-colouring model. The rationale behind this is to get an insight into the behaviour of the algorithm in seeking out clash-free timetables before deciding on the best approach for tackling the more complex problem of conflicting primary and secondary objectives. This is in line with the points made by Ross, Hart and Corne [5] when analysing the reasons for the relatively poor performance of a particular GA approach to the problem. Our intention is to investigate the performance of a family of algorithms based on the ANTCOL family of Costa and Hertz, but with modifications and enhancements, many of which exploit the additional knowledge inherent in the examination-scheduling problem. If successful the more promising performers will then be used as the basis for future work in the development of an ant algorithm that is able to accommodate secondary objectives and additional constraints.

The experiments.

Our experiments use a set of graphs derived from examination scheduling data. Costa and Hertz [2] suggest 8 variants of ANTCOL, all with the same definition of the trail, but using a number of different definitions of visibility based on successful greedy approaches from the graph-colouring literature. Given the conflicting views of Vesel and Zerovnik [6], and the fact that examination scheduling graphs tend to have a different structure to that of randomly generated graphs we repeat some of their experiments on our data and compare the performance of these 8 basic implementations with a series of modifications as follows.

1. A series of different definitions of the quality of a solution used to determine the strength of the trail. In the original implementations the number of colours required is used as a measure of quality. In examination scheduling we usually know the number of timeslots available. This allows a number of different quality measures, for example the number of additional colours needed to colour the graph, or the number of vertices still uncoloured when all the available colour sets are full.
2. A new measure of visibility for the first vertex in each new colour group based on the dummy vertices commonly used to add time-window constraints to the basic graph colouring model.
3. A diversification strategy to encourage the colouring of uncoloured vertices.
4. The use of candidate lists to limit the vertices considered to those with the highest trail factor or highest visibilities.
5. The use of a different trail factor

The ANTCOL family and reasons for introducing each of these modifications will be described in full and the results of the experiments reported. Conclusions as to the best candidates for further development, based on these results together with a discussion as to the suitability of the different underlying greedy strategies for incorporating second order conflict, will be drawn.

References

1. Carter, M. W., Laporte, G. and Chinneck,J. W., A general examination scheduling system, Interfaces 24:3, (1994), pp109-120.

2. Costa, D. and Hertz, A., Ants can colour graphs. Journal of Operational Research

Society, 48, (1997), pp295-305.

3. Maniezzo, V. and Carbonaro, A., An ANTS heuristics for the frequency assignment problem. Future generation Computer Systems 16, (2000), North Holland/Elsevier Amsterdam, pp927-935.

4. Corne, D., Ross, P. and Fang, H. L., Fast practical evolutionary timetabling. Lecture Notes in Computer Science, Vol 865, T. Fogarty (Ed), Springer Verlag, 1994.

5. Ross, P., Hart, E. and Corne, D., Some observations about GA based exam timetabling.. Practice and Theory of Automated Timetabling II (1998) E Burke and M Carter (eds) Springer 115-129.

6. Vesel, A. and Zerovnik, J., How well can ants colour graphs, Journal of computing and information technology - 8, (2000), 2, pp131-136

Appendix 4.2

Sensitivity of α and β with timeslot structure

Statistic definition

Feasible – Number of feasible timetables per 100 created.

Unallocated – Average number of exams not scheduled per timetable.

		Feasible				Unallocated			
Heuristic	*α/β*	*1*	*2*	*3*	*4*	*1*	*2*	*3*	*4*
A	*1*	3.89	2.05	1.85	18.38	0.13	2.65	8.72	1.36
	2	68.98	78.94	87.57	89.07	0.53	0.32	0.18	0.15
	3	90.61	92.52	92.93	94.53	0.20	0.13	0.11	0.08
	4	93.79	77.08	95.36	96.13	0.14	0.29	0.08	0.06
B	*1*	0.00	0.00	0.00	0.00	5.45	5.44	5.41	5.44
	2	48.71	0.46	0.02	0.00	1.10	2.98	3.80	4.03
	3	81.09	40.04	1.15	0.03	0.41	1.03	2.26	3.25
	4	90.53	71.85	19.23	0.32	0.25	0.52	1.34	2.57
C	*1*	0.14	2.52	12.46	23.59	4.06	2.88	1.99	1.49
	2	86.58	91.30	92.38	92.83	0.29	0.18	0.132	0.11
	3	93.68	95.27	96.16	96.45	0.16	0.11	0.08	0.07
	4	94.62	95.74	96.21	96.79	0.13	0.09	0.07	0.06
D	*1*	3.85	12.88	27.04	39.68	2.94	2.02	1.43	1.07
	2	90.27	89.21	92.63	93.42	0.21	0.19	0.14	0.12
	3	96.58	94.73	96.49	96.97	0.10	0.11	0.07	0.06
	4	96.09	58.70	97.18	97.06	0.10	0.46	0.05	0.05
E	*1*	0.00	0.00	0.00	0.00	10.56	10.34	9.65	8.64
	2	0.00	0.00	0.00	0.00	9.42	9.23	8.56	7.99
	3	0.00	0.00	0.00	0.00	9.13	8.72	8.01	7.51
	4	0.00	0.00	0.00	0.00	8.71	8.13	7.23	6.87
F	*1*	0.02	17.89	58.23	67.23	7.77	1.98	1.11	0.86
	2	85.32	86.42	92.71	96.54	0.34	0.29	0.08	0.04
	3	91.72	90.54	95.63	98.56	0.09	0.10	0.05	0.02
	4	96.87	97.45	97.84	99.01	0.04	0.03	0.03	0.02
G	*1*	0.54	1.42	2.98	4.53	3.36	2.90	2.54	2.29
	2	77.72	81.65	81.42	81.72	0.42	0.33	0.31	0.29
	3	89.63	89.46	90.56	89.41	0.20	0.20	0.17	0.18
	4	86.65	92.88	75.76	93.38	0.24	0.24	0.32	0.12
H	*1*	0.00	0.00	0.08	0.33	5.80	4.93	4.15	3.51
	2	0.23	0.96	2.79	5.80	3.90	3.25	2.71	2.25
	3	0.59	1.74	4.49	8.36	3.51	2.95	2.45	2.06
	4	0.77	2.31	5.23	8.95	3.40	2.84	2.38	2.01

Table A4.1 – Average feasible rate per 100 and average unallocated exams per timetable for selected range of α and β across all construction heuristics for HEC data set.

		Feasible Timetables				Unallocated			
Heuristic	*α/β*	*1*	*2*	*3*	*4*	*1*	*2*	*3*	*4*
A	*1*	0.00	0.00	0.052	0.62	9.64	7.18	5.24	3.82
	2	39.05	78.20	85.22	85.82	1.74	0.66	0.40	0.32
	3	74.03	92.45	94.63	93.71	0.79	0.31	0.21	0.17
	4	64.07	91.43	95.07	94.03	0.92	0.29	0.18	0.16
B	*1*	0.00	0.00	0.00	0.00	11.69	11.38	11.23	11.19
	2	6.56	0.00	0.00	0.00	3.78	6.26	7.92	8.94
	3	40.01	0.4	0.00	0.00	1.58	3.36	5.37	6.63
	4	34.11	0.09	0.00	0.00	1.54	3.02	4.27	4.88
C	*1*	0.00	0.00	0.83	8.48	9.79	7.14	4.88	3.18
	2	81.32	89.44	94.10	95.32	0.83	0.43	0.25	0.18
	3	85.24	94.97	96.31	96.73	0.51	0.25	0.17	0.14
	4	94.01	95.10	96.62	97.35	0.33	0.22	0.15	0.12
D	*1*	7.14	1.76	0.73	0.29	0.12	40.54	66.88	84.63
	2	2.41	0.22	0.10	0.07	17.12	94.66	97.17	97.48
	3	2.94	0.16	0.08	0.05	0.05	96.84	98.25	98.74
	4	2.72	0.15	0.07	0.05	0.04	97.21	98.46	98.76
E	*1*	0.00	0.00	0.00	0.00	10.91	10.60	10.53	10.47
	2	0.00	0.00	0.00	0.00	6.75	7.56	8.43	9.92
	3	0.00	0.00	0.00	0.00	7.82	7.66	7.77	7.43
	4	0.00	0.00	0.00	0.00	8.04	8.10	7.17	7.97
F	*1*	0.01	23.82	43.46	51.69	8.14	2.71	1.46	1.07
	2	49.35	93.47	95.15	96.47	1.13	0.28	0.17	0.11
	3	37.42	95.12	97.33	98.15	1.77	0.22	0.13	0.08
	4	0.00	77.61	97.69	98.26	3.45	0.37	0.11	0.08
G	*1*	0.00	0.00	0.02	0.04	7.35	6.52	5.84	5.26
	2	43.27	42.86	48.68	44.15	1.47	1.38	1.28	1.28
	3	58.19	61.29	75.82	76.21	1.13	0.81	0.62	0.56
	4	61.89	80.85	74.02	66.19	0.99	0.52	0.60	0.63
H	*1*	0.00	0.00	0.00	0.00	15.51	13.03	11.01	9.42
	2	0.00	0.03	0.11	0.40	8.31	6.84	5.74	4.99
	3	0.01	0.08	0.31	0.86	6.98	5.93	5.08	4.46
	4	0.01	0.13	0.34	1.00	6.78	5.70	4.95	4.29

Table A4.2 – Average feasible rate per 100 and average unallocated exams per timetable for selected range of α and β across all construction heuristics for EAR data set.

		Feasible Timetables				Unallocated			
Heuristic	*α/β*	*1*	*2*	*3*	*4*	*1*	*2*	*3*	*4*
A	*1*	0.00	0.51	11.52	36.10	7.96	4.56	2.25	1.12
	2	76.38	88.41	92.94	95.15	1.02	0.32	0.14	0.08
	3	92.08	95.44	97.25	97.64	0.39	0.16	0.07	0.05
	4	94.14	96.37	97.90	98.19	0.29	0.13	0.06	0.04
B	*1*	0.00	0.00	0.00	0.00	11.41	11.18	11.17	11.21
	2	53.58	14.78	0.00	0.00	2.63	4.33	7.19	9.29
	3	86.80	82.57	51.98	23.78	0.79	0.92	1.56	2.30
	4	91.92	86.53	73.66	43.33	0.50	0.62	0.89	1.59
C	*1*	0.00	0.60	13.40	39.19	8.40	5.12	2.54	1.28
	2	86.59	92.64	95.87	97.06	0.68	0.26	0.12	0.07
	3	94.42	96.45	97.49	98.38	0.32	0.15	0.08	0.05
	4	95.81	97.21	98.00	98.62	0.25	0.12	0.07	0.04
D	*1*	3.82	50.30	81.88	93.50	4.64	1.08	0.32	0.10
	2	91.88	96.68	98.63	99.30	0.29	0.09	0.03	0.01
	3	76.58	97.74	98.98	99.42	0.38	0.07	0.02	0.01
	4	96.05	98.07	99.08	99.46	0.17	0.06	0.02	0.01
E	*1*	0.00	0.00	0.00	0.00	10.39	9.95	9.58	9.37
	2	12.00	32.60	30.21	0.29	2.65	1.97	3.11	3.91
	3	18.51	52.67	36.60	0.09	1.77	1.77	2.72	4.39
	4	55.66	0.02	18.77	0.15	1.17	2.98	3.58	4.80
F	*1*	0.50	19.95	56.45	77.44	6.56	2.46	0.90	0.40
	2	90.66	95.49	97.75	98.11	0.40	0.15	0.07	0.04
	3	95.61	97.18	98.44	98.90	0.23	0.11	0.05	0.03
	4	77.26	78.53	98.63	98.96	0.39	0.29	0.05	0.03
G	*1*	0.05	0.22	0.96	3.60	6.33	5.26	4.25	2.43
	2	83.39	83.65	83.16	83.76	0.67	0.53	0.49	0.43
	3	92.68	92.60	92.22	88.80	0.32	0.29	0.25	0.28
	4	93.57	92.65	93.19	94.04	0.27	0.25	0.21	0.18
H	*1*	0.00	0.00	0.00	0.00	16.44	13.42	10.89	8.83
	2	0.00	0.02	0.26	1.02	9.59	7.42	5.57	4.73
	3	0.05	0.24	0.88	3.04	7.27	5.96	4.93	3.94
	4	0.09	0.37	1.31	3.65	6.75	5.65	4.57	3.76

Table A4.3 – Average feasible rate per 100 and average unallocated exams per timetable for selected range of α and β across all construction heuristics for TRENT data set.

Appendix 5.1

Static Trail Reinforcement

Results for the HEC data set with upper bounds of 1000, 1250, 1500 and 1750 are as follows:

	UB=1000		UB=1250		UB=1500		UB=1750	
λ	*Average*	*Best*	*Average*	*Best*	*Average*	*Best*	*Average*	*Best*
1	1571.32	824	1571.32	824	1571.32	824	1571.32	824
5	1549.11	841	1272.05	800	1250.22	818	1314.05	769
10	1495.67	796	1333.34	780	1428.69	800	1362.16	783
15	1386.37	834	1142.71	816	1220.77	808	1284.51	813
20	1361.37	798	1188.29	765	1360.60	892	1342.33	833
25	1138.33	753	1287.63	810	1161.70	717	1407.50	822
30	1133.65	739	1230.24	816	1303.35	818	1298.38	848
35	1306.62	743	1239.64	807	1156.29	785	1433.83	830
40	1120.11	797	1216.61	776	1224.81	803	1400.00	834
45	1081.14	796	1312.62	880	1210.01	847	1112.91	709
50	1376.22	832	1024.59	760	1178.63	743	1399.74	863

Table A5.1– Statistics for range of λ on bias term for various UB for HEC

Results for the EAR data set with upper bounds of 1250, 1500, 1750 and 2000 are as follows:

	UB=1250		UB=1500		UB=1750		UB=2000	
λ	Average	Best	Average	Best	Average	Best	Average	Best
1	1650.19	1292	1650.19	1292	1650.19	1292	1650.19	1292
5	1650.19	1292	1642.10	1257	1531.31	1255	1598.40	1191
10	1650.19	1292	1686.89	1371	1599.33	1348	1572.20	1261
15	1650.19	1292	1598.87	1278	1627.20	1348	N/a	N/a
20	1650.19	1292	1632.19	1303	1590.13	1342	1606.24	1247
25	1650.19	1292	1531.01	1205	1591.57	1370	1707.73	1442
30	1650.19	1292	N/a	N/a	N/a	N/a	N/a	N/a
35	1650.19	1292	1549.40	1190	1498.29	1139	1567.95	1319
40	1650.19	1292	1567.04	1200	N/a	N/a	1668.03	1419
45	1650.19	1292	1450.19	1192	N/a	N/a	1571.68	1295
50	1650.19	1292	1581.15	1234	1392.56	1182	1572.57	1244

Table A5.2– Statistics for range of λ on bias term for various UB for EAR

Results for the TRENT data set with upper bounds of 1250, 1500, 1750 and 2000 are as follows:

	UB=1250		UB=1500		UB=1750		UB=2000	
λ	*Average*	*Best*	*Average*	*Best*	*Average*	*Best*	*Average*	*Best*
1	1397.87	1083	1397.87	1083	1397.87	1083	1397.87	1083
5	1414.89	1096	1473.36	1112	1454.54	1084	1525.24	1167
10	1379.34	1043	1411.68	1076	1405.24	1072	1454.02	1143
15	1396.71	1061	1358.29	1068	1412.59	1041	1501.94	1172
20	1382.47	1044	1402.37	1058	1396.33	1048	1529.40	1192
25	1371.03	1003	1368.60	1070	1451.49	1155	1449.89	1090
30	1383.50	1045	1324.20	1060	1425.02	1096	1472.26	1164
35	1348.03	1027	1422.10	1149	1469.51	1141	1444.32	1115
40	1361.85	1022	1355.12	1113	1430.94	1120	1497.15	1211
45	1377.80	1032	1370.66	1143	1444.23	1133	1480.75	1148
50	1376.51	1036	1376.12	1106	1389.13	1078	1498.05	1110

Table A5.3– Statistics for range of λ on bias term for various UB for TRENT

Appendix 6.1

Regression Analysis (TSP)

This section quantifies the relationship between *Before TSP* and *After TSP*. Figures 6.4 – 6.6 depict a positive trend relationship between these variables. For each data set, the scores from all runs were combined and sorted according to *Before TSP* score. This sort was extended to the associated *After TSP* scores. These data were grouped into intervals of size n and averaged to produce 100 data points.

A simple linear relationship is a little spurious due to the differing gradients in each graph, however a series of regression analyses were performed (quadratic, cubic, logarithmic) and simple linear regression accounts for the largest variation (R^2).

		HEC	EAR	TRENT
R^2		0.57	0.84	0.73
Regression Constant		574.19	762.43	936.38
Regression Slope		0.08	0.28	0.16
K-S	*Z*	1.79	0.66	1.15
	Sig	0.00	0.78	0.15

Table A6.1 – Regression analysis of Before TSP versus After TSP

Interpretation

The simple linear regression equations are as follows:

HEC: *After TSP* = 574.19 + 0.08**Before TSP*

For each unit increase in the *Before TSP* score, the *After TSP* score increases by a measure 0.08.

EAR: *After TSP* = 762.43 + 0.28**Before TSP*

For each unit increase in the *Before TSP* score, the *After TSP* score increases by a measure of 0.28.

TRENT: *After TSP* = 936.38 + 0.16**Before TSP*

For each unit increase in the *Before TSP* score, the *After TSP* score increases by a measure of 0.16.

The R^2 values are encouraging and suggest the appropriate use of simple linear regression for the relationship between the variables in question. As Figure 6.4 indicated, the association for HEC is not as strongly linear as the other data sets.

Each of the gradient coefficients is positive, so, highlighting the increasing relationship between *Before TSP* and *After TSP* scores. The intercept (constant) terms are a little misleading. If there are no *Before TSP* second-order clashes then there it is impossible to have any *After TSP* clashes. However, the predicted *After TSP* scores are 574.19, 762.43 and 936.38 for HEC, EAR and TRENT respectively. The regression models have been formed to represent the data sampled and assume that future forecasting/prediction exercises will use *Before TSP* scores near to the interval of data sampled and limited extrapolation will be performed.

After each model was produced residual analysis was performed to check the validity of the model. The One-Sample Kolmogorov-Smirnov Test was used. The statistics of interest are Z and Sig. These are interchangeable. The higher Z, the lower Sig. The closer Sig is to zero, the closer to non-normality the residuals lay. In this section, if Sig>0.05 then we deduce that the residual distribution is normal. The model for EAR has residuals that are soundly normal (Sig=0.78) and TRENT is acceptable to have Normal residuals (Sig=0.15), while HEC is significantly non-Normal (Sig=0.00), which raises doubts regarding the validity of the model.

Figures 6.7-6.9 illustrate the relationship between *Before TSP* and *Savings*. The former variable refers to the *Before TSP* adjustment second-order score while the latter relates to the number of second-order conflicts removed from a timetable. Once again the data were sorted and averaged into 100 intervals. The least squares statistics are as follows:

		HEC	EAR	TRENT
R^2		1	0.97	0.99
Regression Constant		-574.19	-762.43	-936.38
Regression Slope		0.92	0.72	0.84
K-S	*Z*	1.79	0.66	1.15
	Sig	0.00	0.78	0.17

Table A6.2 – Regression analysis of Before TSP versus Savings

Interpretation

The simple linear regression equations are as follows:

HEC: *Savings* = -574.19 + 0.92**Before TSP*

For each unit increase in the Before TSP score, the Savings score increases by a measure 0.08.

EAR: *Savings* = -762.43 + 0.72**Before TSP*

For each unit increase in the Before TSP score, the Savings score increases by a measure of 0.28.

TRENT: *Savings* = -936.38 + 0.84**Before TSP*

For each unit increase in the *Before TSP* score, the *Savings* score increases by a measure of 0.84.

Firstly, we should note the similarity in the statistics between *Savings* and *After TSP* models. The absolutes of each of the constants are the same and adding each slope would equal 1 (i.e. HEC, 0.92+0.08=1). This is to be expected since these models derive from the same data sets. The most notable outcome of this study is the explained variation. The R^2 statistics suggest perfect linearity.

Appendix 6.2

After TSP second-order statistics when using min+deviation rule

Run	*% Dev Run Min*	*Av*	*Best*	*Best 10*	*Worst 10*	*Eligible*	*Time*	*% Dev Best*	*% Dev Time*
	Infinite	687.16	574	582.20	850.80	7995	596.33	0	0
	0	677.05	579	647.50	707.00	19	85.07	0.87	85.73
	1	647.62	579	586.00	717.70	277	101.61	0.87	82.96
	3	654.20	579	584.20	740.90	1249	163.91	0.87	72.51
1	*5*	658.16	574	582.20	752.30	1746	195.77	0	67.17
	10	663.13	574	582.20	772.70	2286	230.38	0	61.37
	20	670.99	574	582.20	806.30	3205	289.29	0	51.49
	30	673.82	574	582.20	827.70	3733	323.14	0	45.81
	50	677.45	574	582.20	833.80	4227	354.80	0	40.50
	Infinite	637.21	557	570.90	795.00	7604	486.66	0	0
	0	640.00	584	617.70	661.50	18	84.49	4.85	85.18
	1	631.98	579	588.40	699.70	234	98.33	3.95	82.74
	3	629.83	579	581.60	707.80	1103	154.04	3.95	72.98
2	*5*	629.96	578	580.40	714.90	2172	222.56	3.77	60.95
	10	632.52	571	574.70	737.20	4337	361.34	2.51	36.61
	20	634.79	571	574.40	751.90	6870	523.70	2.51	8.12
	30	635.21	571	574.40	757.10	7094	538.06	2.51	5.60
	50	636.08	557	572.50	765.30	7305	551.58	0	3.23
	Infinite	629.45	512	523.60	793.00	7765	580.23	0	0
	0	625.13	575	596.30	657.90	24	84.03	12.30	85.52
	1	617.07	561	572.10	689.20	102	89.03	9.57	84.66
	3	612.64	550	560.80	702.80	314	102.62	7.42	82.31
3	*5*	612.84	539	551.50	713.70	686	126.47	5.27	78.20
	10	611.35	512	525.90	724.00	1999	210.63	0	63.70
	20	620.48	512	523.60	741.90	4014	339.79	0	41.44
	30	624.44	512	523.60	756.90	5861	458.18	0	21.03
	50	628.38	512	523.60	764.70	7420	558.16	0	3.81
	Infinite	595.61	520	531.50	762.00	7355	552.36	0	0
	0	614.54	561	588.90	62.30	13	81.74	7.88	85.20
	1	575.14	526	539.90	668.20	901	138.66	1.15	74.90
	3	578.45	526	539.10	682.50	1610	184.11	1.15	66.67
4	*5*	583.62	526	538.50	687.30	4121	345.06	1.15	37.52
	10	586.89	526	537.00	689.60	5128	409.61	1.15	25.84
	20	592.15	520	532.90	706.30	6504	497.81	0	9.88
	30	593.29	520	532.90	706.80	6855	520.31	0	5.80
	50	594.08	520	532.00	718.20	7092	535.50	0	3.05
	Infinite	687.50	580	582.10	846.00	7982	594.79	0	0
	0	681.82	617	653.40	712.50	17	84.23	6.38	85.84
	1	697.59	617	635.30	742.60	111	90.26	6.38	84.83
	3	681.02	592	603.10	761.30	425	110.39	2.07	81.44
5	*5*	689.23	592	601.50	787.50	838	136.86	2.07	76.99
	10	690.95	580	591.50	802.70	1905	205.25	0	65.49
	20	688.95	580	588.30	809.70	3619	315.12	0	47.02
	30	685.77	580	583.80	813.70	4741	387.04	0	34.93
	50	685.10	580	583.20	821.40	5315	423.84	0	28.74

Table A6.3 – Statistics describing After TSP results for min+deviation rule for HEC.

Run	% Dev Run Min	Av	Best	Best 10	Worst 10	Eligible	Time	% Dev Best	% Dev Time
	Infinite	1149.03	919	937.90	1460.20	6866	1147.57	0	0
	0	1105.87	973	1046.20	1168.10	23	377.40	5.88	67.11
	1	1087.18	944	995.60	1191.40	55	381.00	2.72	66.80
	3	1084.51	944	978.60	1217.90	187	395.86	2.72	65.50
1	*5*	1089.61	936	955.10	1259.80	588	440.99	1.85	61.57
	10	1102.14	936	946.10	1308.40	2187	620.95	1.85	45.89
	20	1132.88	935	941.10	1404.30	5148	954.21	1.74	16.85
	30	1144.38	919	937.90	1429.00	6457	1101.54	0	4.01
	50	1148.86	919	937.90	1443.10	6859	1146.78	0	0.07
	Infinite	1175.43	990	1004.90	1464.00	7453	1195.99	0	0
	0	1128.40	1038	1078.50	1178.3	20	359.41	4.85	69.95
	1	1136.55	1038	1069.80	1214.7	29	360.43	4.85	69.86
	3	1136.66	1031	1046.40	1248.4	83	366.50	4.14	69.36
2	*5*	1141.85	1031	1042.70	1288	176	376.97	4.14	68.48
	10	1156.63	1019	1032.30	1379.5	705	436.51	2.93	63.50
	20	1168.06	990	1005.60	1432.9	3189	716.08	0	40.13
	30	1173.14	990	1005.10	1444.5	5824	1012.65	0	15.33
	50	1175.36	990	1004.90	1464	7429	1193.29	0	0.23
	Infinite	1152.78	948	963.10	1432.70	5793	990.47	0	0
	0	1133.29	1019	1092.60	1168.40	14	340.05	7.49	65.68
	1	1113.23	970	1038.00	1192.40	30	341.85	2.32	65.49
	3	1114.79	970	1001.20	1269.00	133	353.44	2.32	64.32
3	*5*	1114.86	970	983.50	1286.10	447	388.78	2.32	60.75
	10	1131.29	948	965.60	1346.90	2640	635.60	0	35.83
	20	1150.13	948	963.10	1424.90	5424	948.94	0	4.19
	30	1152.57	948	963.10	1432.70	5769	987.77	0	0.27
	50	1152.78	948	963.10	1432.70	5789	990.02	0	0.05
	Infinite	1131.01	903	926.30	1385.60	6091	1087.83	0	0
	0	1069.29	991	1044.00	1092.80	14	403.87	9.75	62.87
	1	1078.12	991	1014.00	1151.40	33	406.01	9.75	62.68
	3	1080.66	977	993.60	1191.50	142	418.28	8.19	61.55
4	*5*	1090.42	958	979.70	1216.40	415	449.00	6.09	58.73
	10	1107.25	905	939.40	1273.20	2168	646.30	0.02	40.59
	20	1124.14	903	926.30	1332.20	5175	984.74	0	9.48
	30	1130.51	903	926.30	1374.20	5987	1076.12	0	1.08
	50	1130.94	903	926.30	1385.60	6087	1087.38	0	0.04
	Infinite	1045.52	842	863.80	1322.60	5417	1100.65	0	0
	0	991.63	895	948.20	1034.30	16	492.77	6.29	55.23
	1	1017.93	895	930.20	1118.70	30	494.35	6.29	55.09
	3	1007.50	868	912.40	1156.60	104	502.68	3.09	54.33
5	*5*	1009.18	868	896.80	1182.80	324	527.44	3.09	52.08
	10	1025.82	852	879.90	1211.20	1585	669.36	1.12	39.18
	20	1042.89	842	863.80	1252.30	4761	1026.82	0	6.71
	30	1044.89	842	863.80	1298.00	5366	1094.91	0	0.52
	50	1045.44	842	863.80	1322.60	5414	1100.31	0	0.03

Table A6.4 – Statistics for After TSP results for dynamic min+deviation rule for EAR.

Run	% Dev Run Min	Av	Best	Best 10	Worst 10	Eligible	Time	% Dev Best	% Dev Time
	Infinite	1155.22	930	964.40	1361.50	6710	1360.84	0	0
	0	1142.78	1046	1093.30	1192.20	23	631.30	12.47	53.61
	1	1138.75	1028	1065.10	1213.10	44	633.59	10.54	53.44
	3	1134.33	999	1046.80	1238.00	134	643.41	7.42	52.72
1	*5*	1134.57	999	1017.80	1258.70	370	669.16	7.42	50.83
	10	1143.93	989	1007.10	1304.90	1782	823.21	6.34	39.51
	20	1153.71	930	971.40	1351.70	5040	1178.65	0	13.39
	30	1154.72	930	964.40	1361.50	6448	1332.26	0	2.10
	50	1155.21	930	964.40	1361.50	6707	1360.51	0	0.02
	Infinite	1105.17	942	966.50	1338.0	7259	1409.27	0	0
	0	1092.04	999	1034.60	1153.0	26	620.16	6.05	55.99
	1	1100.86	999	1026.40	1190.5	44	622.13	6.05	55.85
	3	1096.81	969	991.70	1250.0	154	634.13	2.87	55.00
2	*5*	1091.80	969	989.00	1256.9	355	656.06	2.87	53.45
	10	1101.19	969	982.40	1300.0	1438	774.21	2.87	45.06
	20	1106.84	963	974.30	1324.2	4160	1071.18	2.23	23.99
	30	1106.73	942	969.40	1333.3	6062	1278.68	0	9.27
	50	1105.19	942	966.50	1338.0	7220	1405.02	0	0.3
	Infinite	1152.18	965	985.10	1358.80	7120	1399.40	0	0
	0	1126.05	1016	1060.10	1190.80	21	625.04	5.29	55.34
	1	1121.79	1016	1036.60	1209.00	34	626.46	5.29	55.23
	3	1110.95	1016	1029.20	1226.30	110	634.75	5.18	54.64
3	*5*	1105.67	1015	1026.80	1255.90	281	653.41	3.42	53.31
	10	1117.71	998	1015.70	1299.00	1555	792.40	0	43.38
	20	1144.11	965	988.30	1337.20	5073	1176.20	0	15.95
	30	1150.90	965	985.10	1355.90	6714	1355.23	0	3.16
	50	1152.16	965	985.10	1358.80	7105	1397.89	0	0.10
	Infinite	1136.01	953	970.90	1350.30	7258	1410.80	0	0
	0	1086.43	986	1028.60	1144.10	21	621.26	3.46	55.96
	1	1092.91	986	1015.20	1192.30	44	623.77	3.46	55.79
	3	1080.27	985	1001.60	1207.10	86	628.35	3.36	55.46
4	*5*	1093.12	957	993.00	1235.10	172	637.73	0.42	54.80
	10	1106.27	957	982.10	1262.80	789	705.05	0.42	50.03
	20	1127.00	953	971.20	1307.30	3588	1010.41	0	28.38
	30	1133.86	953	970.90	1331.20	5965	1269.74	0	10.00
	50	1135.88	953	970.90	1350.30	7174	1401.64	0	0.60
	Infinite	1129.46	956	970.00	1365.60	7035	1387.51	0	0
	0	1098.81	990	1028.80	1170.60	21	622.30	3.56	55.15
	1	1095.79	990	1003.50	1227.40	43	624.70	3.56	54.98
	3	1083.73	990	998.90	1251.30	137	634.95	3.56	54.24
5	*5*	1082.81	990	996.30	1261.50	332	656.23	3.56	52.70
	10	1101.60	983	990.90	1292.30	1425	775.47	2.82	44.11
	20	1122.81	956	975.80	1350.20	4400	1100.04	0	20.72
	30	1127.94	956	970.00	1358.20	6089	1284.30	0	7.43
	50	1129.41	956	970.00	1365.60	7002	1383.91	0	0.26

Table A6.5 – Statistics for After TSP for dynamic min+deviation rule for TRENT.

Appendix 6.3

Local Search

Let us consider the objective of minimizing a function $F(x)$ for a discrete optimization problem, where x is a member of X, the finite set of solutions. A neighbourhood structure $N(x)$ is defined as the set of solutions which are adjacent to any solution $x \in X$. Local search begins from a solution s and another solution, s^1, from the neighbourhood of s is generated, usually randomly. These solutions are then compared and if $F(s^1)<F(s)$ then s^1 replaces s, otherwise s remains. This is known as random descent and is defined as follows.

1. Select $s^1 \in X$
2. For steps $n=1, 2, \ldots, s^n$ let s^n denote the current solution
    a. Choose randomly s^1 from neighbourhood $N(s^n)$
    b. If $F(s^1)<F(s^n)$ then $s^{n+1}=s^1$; else $s^{n+1}=s^n$
    c. If there no improvement is observed for n steps then stop.

Steepest Descent has been used in Section 6.3.1 due to the superior solution quality over Random Descent. Steepest Descent differs from Random Descent since it evaluates the cost function value of every neighbour. The neighbour producing the largest decrease in the cost function is accepted as the new solution and the search continues until the neighbourhood contains no improving moves.

Local search is relatively simple to implement with four factors that have to be deduced: solution space X, starting solution s^1, cost function $F(x)$ and the neighbourhood structure $N(x)$. The manner in which these factors are defined has a major influence on solution quality. With respect to the implementation of Steepest Descent Strategy in Section 6.3, s^1 represents the ant-based (*Before Loc*) solution, F(x) refers to the second-order score of the current solution x, $N(x)$ is the set of feasible solutions that can be reached from current solution x through changing the timeslot of one exam. The solution space X refers to all combinations of examination timetables.

Appendix 6.4

Regression Analysis (Steepest Descent)

Appendix 6.4 mirrors the study of Appendix 6.1. On this occasion, the results generated from Steepest Descent (Section 6.3.1) are used. We attempt to quantify the relationship between *Before Loc* with *After Loc/Savings.*

The statistics for the relationship between Before Loc and After Loc are as follows:

		HEC	EAR	TRENT
R^2		0.31	0.99	0.99
Regression Constant		526.11	453.66	226.14
Regression Slope		0.21	0.46	0.58
K-S	*Z*	1.99	0.99	1.62
	Sig	0.00	0.28	0.01

Table A6.6 – Regression analysis of Before Loc versus After Loc

Interpretation

HEC: *After Loc* = 526.11+0.21**Before Loc*

For each unit increase in the *Before Loc* score, the *After Loc* score increases by 0.21.

EAR: *After Loc* = 453.66+0.46**Before Loc*

For each unit increase in the *Before Loc* score, the *After Loc* score increases by 0.46.

TRENT: *After Loc* = 226.14+0.58**Before Loc*

For each unit increase in the *Before Loc* score, the *After Loc* score increases by 0.58.

The R^2 measures for EAR and TRENT are strong, while the same statement cannot be made for HEC. The positive slope highlights that better *Before Loc* scores encourage better *After Loc* scores. This is a generalized interpretation and cannot account for 'odd' good *After Loc* scores that have derived from relatively poor *Before Loc* timetables.

The HEC and TRENT models do not observe the assumptions regarding normality of residuals. The Sig values suggest clear non-Normality.

The same study is performed to describe the relationship between Before Loc and Savings. The regression statistics are tabulated below:

		HEC	EAR	TRENT
R^2		0.86	0.99	0.97
Regression Constant		-526.11	-453.66	-226.14
Regression Slope		0.79	0.54	0.42
K-S	*Z*	1.99	0.99	1.62
	Sig	0.00	0.28	0.01

Table A6.7– Regression analysis of Before Loc versus Savings

Interpretation

HEC: *Savings* = -526.11+0.79**Before Loc*

For each unit increase in the *Before Loc* score, the *Savings* score increases by 0.79.

EAR: *Savings* = -453.66+0.54**Before Loc*

For each unit increase in the *Before Loc* score, the *Savings* score increases by 0.54.

TRENT: *Savings* = -226.14+0.42**Before Loc*

For each unit increase in the *Before Loc* score, the *Savings* score increases by 0.42.

As with the parallel study in Appendix 6.1, the R^2 values are strong and indicate the predictive power of the variable *Savings* given *Before TSP*. Despite this, the normality assumptions are not observed on two counts, thus placing doubts on the models. As expected, each model is the inverse of the mirrored model in the *Before Loc/After Loc* study. This is due to the derivation of both model types being from the same original variables.

A note on Normality

For the purpose of this study, the validity of each model is not essential. If these analyses were used for any future investigation then regression assumptions would have to be observed. The main reason behind these appendices is to quantify the relationships between the variables discussed so the reader can visualize the potential importance of encouraging the ants to visit profitable areas of the solution space.

Appendix 6.5

Before Loc second-order statistics for Baldwinian and Lamarckian theories

Run	*% Dev Run Min*	*Av*	*Best*	*Best 10*	*Worst 10*
BALDWIN	Infinite	1048.42	611	644.52	2427.58
	0	1045.95	667	716.22	2282.04
	1	1033.68	667	707.36	2296.48
	3	983.66	652	695.30	2409.04
	5	1040.10	634	690.14	2372.34
	10	1029.47	636	696.66	2381.18
	20	1056.73	660	667.62	2419.80
	30	1031.69	628	682.38	2398.62
	50	997.55	580	639.00	2360.10
LAMARCK	Infinite	1102.76	610	663.62	2403.10
	0	1063.16	585	719.20	2373.68
	1	1005.88	594	704.02	2345.92
	3	1001.54	668	696.68	2326.56
	5	1035.95	663	691.70	2308.28
	10	1008.39	617	674.20	2414.84
	20	1007.47	639	671.72	2381.76
	30	1165.29	639	701.70	2424.28
	50	1047.60	658	719.56	2328.80

Table A6.8 –Before Loc second-order statistics for Baldwinian and Lamarckian theories for HEC

Run	*% Dev Run Min*	*Av*	*Best*	*Best 10*	*Worst 10*
BALDWIN	Infinite	1225.66	950	1017.58	1748.80
	0	1336.24	1027	1130.40	1786.74
	1	1357.06	1023	1116.16	1792.24
	3	1298.97	982	1075.10	1810.04
	5	1268.05	1004	1083.62	1784.54
	10	1219.36	975	1014.44	1787.78
	20	1273.79	962	1056.08	1793.22
	30	1237.24	910	1015.14	1793.28
	50	1217.43	912	1005.84	1785.08
LAMARCK	Infinite	1241.08	935	999.00	1807.14
	0	1307.67	961	1084.14	1782.48
	1	1308.56	945	1089.44	1800.38
	3	1262.18	975	1050.40	1801.60
	5	1248.71	924	1038.08	1801.92
	10	1235.39	996	1030.16	1791.66
	20	1222.61	907	1008.00	1788.22
	30	1221.16	953	994.14	1835.02
	50	1228.02	950	1016.12	1802.18

Table A6.9–Before Loc second-order statistics for Baldwinian and Lamarckian theories for EAR

Run	*% Dev Run Min*	*Av*	*Best*	*Best 10*	*Worst 10*
BALDWIN	Infinite	1204.11	894	963.84	1792.60
	0	1341.90	1020	1078.20	1813.50
	1	1306.42	997	1052.04	1803.08
	3	1240.19	908	981.40	1793.80
	5	1244.06	923	988.96	1805.08
	10	1172.98	884	935.90	1810.96
	20	1182.19	881	944.06	1785.56
	30	1202.59	897	960.14	1780.78
	50	1190.12	910	949.08	1791.72
LAMARCK	Infinite	1189.43	904	960.04	1767.02
	0	1300.83	990	1051.76	1783.40
	1	1310.20	985	1045.76	1791.68
	3	1245.66	921	1006.26	1790.98
	5	1205.54	917	962.30	1782.00
	10	1166.90	886	926.24	1780.70
	20	1201.71	847	952.26	1771.04
	30	1200.08	931	957.80	1768.28
	50	1189.29	879	891.60	1762.88

Table A6.10–Before Loc second-order statistics for Baldwinian and Lamarckian theories for TRENT

Appendix 6.6

Relationship between computational effort and improvement

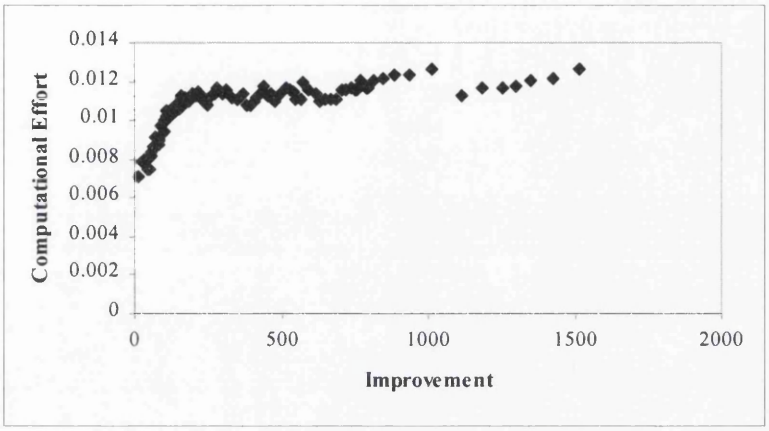


Figure A6.1 – Relationship between Computational Effort (secs) and Improvement (students) for HEC.

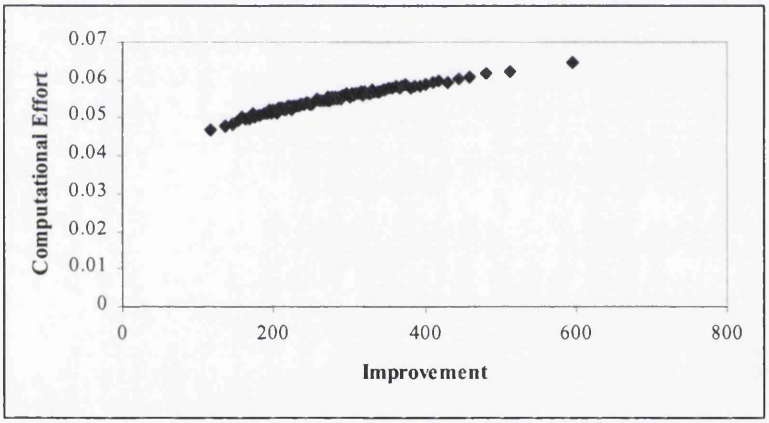


Figure A6.2 – Relationship between Computational Effort (secs) and Improvement (students) for EAR.

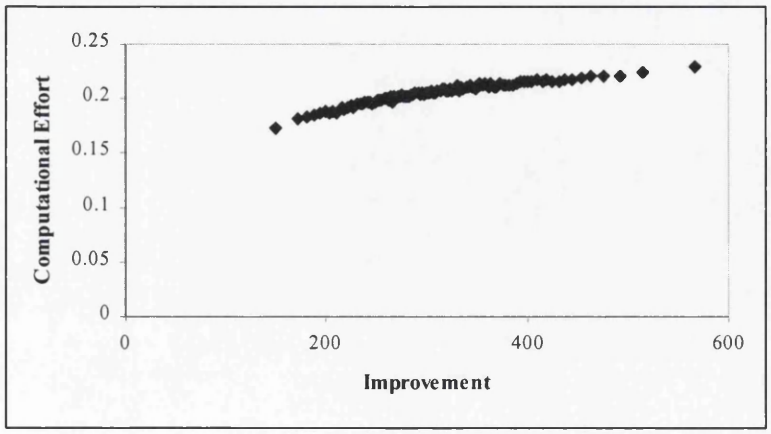


Figure A6.3 – Relationship between Computational Effort (secs) and Improvement (students) for TRENT.

Appendix 6.7

Simulated Annealing

Simulated Annealing, see Eglese (1990) for a detailed account, is a local search technique that searches a wider part of the solution space than Steepest or Random Descent, thus increasing the probability of observing superior solutions. It enables the search process to escape from local minima.

Origins

The principles behind Simulated Annealing stem from statistical thermodynamics theory, where, annealing cools a melted body in a controlled manner and the cooling rates are varied such that the structure of the solid is altered. Metropolis et al. (1953) modeled this process by examining the internal atomic structure of such bodies and the manner in which atoms rearrange as the state changes from melted and high energy (atoms move freely with respect to one another), to solid and low energy (atoms lack movement). A slow rate cooling process results in a minimal energy solid but faster cooling produces suboptimal atomic structures.

During the cooling process, some inferior atomic re-arrangements, which have a higher energy than previous arrangements, are visited. Metropolis et al. simulated this observation using a conditional probability function defined as

$$p(\partial) = \exp^{\frac{-(F(s^1)-F(s))}{kt}} \qquad \text{(A6.1)}$$

where $p(\delta)$ is the probability of accepting the inferior atomic arrangement

$F(s)$ and $F(s^1)$ are the energies of state before and after the proposed change

k represents the Boltzmanns constant

t is the temperature of the system.

Kirkpatrick et al. (1983) and Cěrny (1985) used this an analogy for an optimization technique. The solution space of a problem represents all possible arrangements of atoms.

Each solution has a corresponding energy to its cost function. A small alteration is made to the current solution and the resulting solution is accepted if it generates a decrease in the cost function. If an increase is observed, then acceptance is based probabilistically according to Metropolis criterion $p(\delta)$, which when calculated, is compared with a value, v, with $v \in [0,1]$. If $v<p(\delta)$ then the move is accepted, otherwise there is no change to the solution.

When using SA, Boltzmanns constant can be removed from Metropolis criterion and the parameter t is lowered during the run. This lowers the probability of accepting inferior solutions. Parameter t is lowered until no uphill (worsening) moves are likely to be accepted.

There is a trade-off between solution time and quality. The cooling rate is required to be slow enough to achieve good solutions but quick enough to avoid the execution of an exorbitant number of iterations.

Usually, the final solution may not be the best solution found during the run as the search may become trapped in a relatively poor local optimum.

Appendix 7.1

Derivation of $\delta_{3,d}$

Derivation of $\delta_{3,d}$ values used in Section 7.1.

$$\delta_{3,d} = \frac{750}{Av\ Sec\_score} \times Av_{bench,d} \quad \text{(A7.1)}$$

Av Sec_score refers to the *Average* second-order score across all data sets (Chapter 6). $Av_{bench,d}$ refers to the *Average Carter Cost* across the available benchmarks for all data sets.

Equation A7.1 is formed through the following. A generic setting $\delta_2 = 750$ is used and this is applied to all data sets, so we assume that the average condition of each timetable is approximately the same, hence the presence of *Av Sec_score* in Equation A7.1. The ratio of these values are scaled by the *Average* benchmark *Carter Cost* per data set *d*.

	Merlot et al.	Carter et al.	Di Gaspero et al.	White & Xie	Average Performance	Average Second-Order	$\delta_{3,d}$
C91	5.20	8.38	6.50	-	6.69	2283.35	2.20
C92	4.40	7.04	5.60	4.70	5.44	2075.82	1.96
EAR	35.40	40.92	46.70	-	41.01	955.18	32.20
HEC	10.70	15.04	12.60	-	12.78	657.01	14.59
KFU	14.00	18.76	19.50	-	17.42	1407.29	9.28
LSE	11.00	12.36	15.90	-	13.09	757.58	12.96
STA	157.40	167.14	166.80	-	163.78	3041.35	40.39
TRE	8.60	10.78	10.50	-	9.96	1034.76	7.22
UTA	3.60	4.80	4.50	4.00	4.23	2123.63	1.49
UTE	25.20	30.78	31.30	-	29.09	1055.75	20.67
YOR	37.90	45.60	42.10	-	41.87	804.95	39.01

Table A7.1 – Estimates of $\delta_{3,d}$