Transactions on Industrial Informatics

Manuscript ID TII-20-5532

Comparison of a bat and genetic algorithm generated sequence against lead through programming when assembling a PCB using a 6 axis robot with multiple motions and speeds

C.A. Griffiths, C. Giannetti, K.T. Andrzejewski, A. Morgan,

Abstract— **An optimal component feeder arrangement and robotic placement sequence are both important for improving assembly efficiency. Both problems are combinatorial in nature and known to be NP-hard. This paper presents a novel discrete hybrid bat-inspired algorithm for solving the feeder slot assignment and placement sequence problem encountered when planning robotic assembly of electronic components. In our method, we use the concepts of swap operators and swap sequence to redefine position and velocity operators from the basic bat algorithm. Furthermore, we propose an improved local search method based on genetic operators of crossover and mutation enhanced by the 2-opt search procedure. The algorithm is formulated with the objective of minimizing the total traveling distance of the pick and place device. Through numerical experiments, using a real PCB assembly scenario, we demonstrate the considerable effectiveness of the proposed discrete Bat Algorithm (BA) to improve selection of feeder arrangement and placement sequence in PCB assembly operations and achieve high throughput production. The results also highlighted that the even though the algorithms out performed traditional lead through programming techniques, the programmer must consider the influence of different robot motions and speeds.**

Keywords— Assembly Optimization; Printed Circuit Board; Feeder Slot Assignment; Sequencing Problem; Genetic Algorithm, Discrete Bat Algorithm.

1. Introduction

Implementation of automated assembly of electronics Printed Circuit Board (PCB) offers some distinct advantages over manual methods due to its reliability, flexibility and ability to handle high production demand [1, 2]. Since assembly of electronic components is a complicated task, several technologies that utilize sophisticated machines are used in industry to perform intricate operations [2]. Automated assembly system usually comprises of several sub-systems, those include: part feeding system, work holding and pick and place devices.

In order to take full advantage of automated machines, assembly processes such as component placement require further optimisation to determine the optimal assembly configuration [3]. In the design of an efficient assembly system, it is necessary to determine the optimal (or near-optimal) assembly operations planning, including: i) allocation of components to machines; ii) component feeder allocation; iii) component sequence allocation and iv) placement of pick and place machine [2, 3]. Furthermore, a decision must be made regarding the criterion in which the production performance should be optimized. In the literature, one of the most commonly considered criteria is makespan minimization or, in the context of repetitive assembly, cycle time minimization [4-6].

When minimising cycle-time, component placement sequencing is considered a bottleneck of PCB assembly, hence finding the optimal solution to this problem can yield significant improvements in real situations, especially for assembly of complex board designs [7]. The sequence allocation problem for PCB assembly has been widely studied in the literature with early work introduced by Ball & Magazine [8] and Ahmadi & Mamer [9]. Mathematically, the sequence allocation problem can be formulated as a variant of the travel salesman problem and it requires the use of heuristic algorithms to find efficient solution for complex designs with high number of components due to its combinatoric nature and the fact that is a NP-complete problem. In the literature, several approaches have been proposed to solve this problem using evolutionary programming [10-13], integer programming [14], particle swarm optimisation [15] and Bees algorithm [1, 16]. The need for production efficiency and flexibility faced by PCB manufactures requires continuous research on best optimisation approaches alongside technological advances.

In a recent paper, a novel methodology using an articulated 6-axis robot for assembly of PCB components has been proposed [17]. The authors employ a genetic algorithm (GA) to reduce cycle time for robot path motions. Further research investigated optimisation issues for pick and place operations of multi-gripper robots [18], proposing an intelligent system that combines an efficient swarm intelligence algorithm based on Ant System and Tabu Search with reinforcement learning to reduce cycle time and energy consumption.

PCB manufacture would traditionally be made using high investment production automation equipment. Since the use of articulated robots is becoming more popular in prototype

electronic assembly due to improved accuracy, flexibility and low cost, the development and evaluation of new algorithms to optimise their operations is very desirable for electronic board manufacturer. Basic Lead through (LT) programming is the most common method way to plan robot paths, but as has been identified, heuristic approaches have been effective in optimising sequences. However, consideration of multiple modes of robot motion and robot speed is often overlooked and there is a need to factor in these motions to fully optimise a sequence of operations.

In this paper, we study and evaluate the application of a modified version of a hybrid discrete bat algorithm (BA) to find an optimal PCB components sequence allocation that reduces cycle time, using a highly flexible articulated 6-axis robot. The proposed discrete version of the BA is validated using a KUKA KR16 robot and considers four different movement types. The performance of the BA is compared against traditional GA and LT programming approaches together with four KUKA movement types and speeds. The research offers further insights on suitability of best optimisation practices for reducing cycle time in assembly processes.

2. Literature Review

Optimisation of PCB assembly processes has been widely studied in the literature with focus on different SMT and different aspects of the process optimisation. Since the determination of optimal component sequencing and feeder-slot allocation are considered crucial factors for improving the efficiency of assembly operation in electronic manufacturing, many authors have focussed on solving one or both problems. Ball & Magazine [8] assumed a fixed feeder arrangement and solved the placement sequencing problem with an heuristics approach, modelling it as a rural postman problem, while Ahmadi and Mamer [9] proposed an heuristic approach to solve the component allocation and sequencing problem as a collection of interdependent traveling salesman problems. Grunow et al. [19] applied an heuristic method to optimise operations of a pick-and-place manipulator with a revolving head, modelling the problem as a Vehicle Routing Problem.

In the literature, many authors have considered both the feeder arrangement and component sequence allocation problem trying to solve them simultaneously through integrated approaches. Ellis et al. [21] developed a heuristic approach for solving the feeder arrangement problem and the placement sequencing problem simultaneously for a turret-type machine. Their solution consists of a set of rules to generate an initial component placement sequence and feeder arrangement along with an improvement procedure to improve the initial solution. The effectiveness of the approach was demonstrated on a Fuji CP4-3 machine and real PCB data. Deo et al [20] proposed an approach based on GA for simultaneously optimizing component placement sequence and feeder assignments in the assembly of PCBs. Ho & Ji [22] developed a hybrid GA to minimise the total travelled distance for a sequential pick-and-place machine. The algorithm was tested on a PCB with 200 components and 10 components type, showing efficient performance. Kulak et al [23] proposed an integrated approach using a novel GA to solve feeder assignment and component sequencing for commonly used collect-and-place machines with a revolver-type placement head. Their approach integrated a clustering algorithm for generating sub-sections of the PCB and grouping the corresponding placement operations. To solve this allocation problem, two different heuristic strategies were proposed and detailed numerical experiments were carried out to evaluate the performances of the proposed GAs. Li et al. [24] modelled the sequencing placing problem for a multi-head surface mounting machine as a travel salesmen problem and used a GA to find the optimal feeder allocation [17].

The survey indicates that approaches based on GAs are widely used by many researchers, showing state-of-the-art result on a benchmark for the problem. In more recent times, Swarm–based optimisation algorithms (SOAs) have also emerged. SOAs are metaheuristic nature-inspired algorithms that that mimic swarm behaviour in order to solve optimisation problems. These methods have shown excellent performance when applied to a range of different assembly processes optimisation problems, including the PCB sequence allocation problem. Castellani et al. [1] further improved this method by developing a new problem-specific implementation of the Bees Algorithm with five new operators to simultaneously minimise assembly time and optimise feeder arrangement using a machine of the moving-board-with-time-delay type [25]. Nilakantan et al [3] proposed a model to minimise simultaneously cycle time and energy consumption using a particle swarm optimiser for a robotics assembly line. Parallel Pick-and-Place (PPNP) optimisation was studied in [18]. The authors also used swarm intelligence algorithm based on Ant System and Tabu Search to find an optimal routing and configuration of assembly pickup and placement positions.

Among metaheuristic nature inspired algorithms, the BA is an emerging approach that was proposed for the first time by Yang [26]. The basic BA is based on the use of echolocation characteristics of microbats. Echolocation behaviour of microbats was formulated in a way that when associated with an objective function provides an effective optimisation algorithm. BA, in its original form, was proposed for solving mainly continuous optimization problems. And indeed, many experimental results reported by researchers [27-30] have demonstrated its superior performance and ability to compete with other metaheuristic optimisation algorithms. In industrial and production engineering, BA was implemented by Kaven & Zakian [28] who used it for size optimisation of skeletal structures consisting of trusses and frames. Various optimization problems comprising size, shape, and topology were implemented to demonstrate the ability of the present enhanced BA. An implementation of BA for solving the multistage, multimachine and multiproduct scheduling problem was proposed in [31]. The authors' aim was to minimize both the earliness and tardiness penalties cost, correctly sequence the operations required to manufacture components, and to satisfy

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2021.3082877, IEEE Transactions on Industrial Informatics

the assembly precedence relationship. Experimentally, it was found that the quality of the solutions obtained from BA based scheduling tool can be improved significantly after applying the necessary parameter setting identified by the statistical tools. The results obtained using BA optimized settings outperformed those using a non-optimized setting. Research has been carried out using BA with the intention of addressing real-world discrete optimization problems. Osaba et al. [29] present a discrete version of the BA to solve the well-known symmetric and asymmetric Traveling Salesman Problems. They propose an improvement in the basic structure of the classic BA and compared its performance in 37 instances with the results obtained by several different optimisation techniques. They report that the presented improved BA outperforms significantly all the other alternatives in most of the cases. Similar findings were reported by Amara et al. [27] who proposed as a new concept of Swap Operator and Swap Sequence to redefine respectively BA position and velocity operators for Traveling Salesman Problem. They have also redefined the local search procedure with genetic crossover operator and 3-opt search algorithm. Saji and Riffi [30] have also validated the performance of BA in benchmark to solve the symmetric TSP problem which they achieved by adapting the 2-arc crossover method to enhance their local search. Their computational results agree with the other researchers studying the BA for discrete TSP. A study by Charkri et al. [32] on continuous optimisation problems compared the BA with ten other algorithms using non-parametric statistical tests. The statistical test results show the superiority of the directional bat algorithm. Studies on shortest path planning using a BA by Lijue et al. [33] found that the BA outperforms the alternatives in most cases. In order to further improve robot path efficiency, Saraswathi et al. [34] combined the best qualities of the cuckoo-search algorithm and the BA and found that efficiency in reaching the target was increased when compared to individual algorithms

As shown in the literature review, a plethora of approaches were proposed by scholars for solving sequence component placement problems for component assembly. However, there are relatively few studies that focus on PCB assembly sequence optimisation for articulated 6-axis robots. These robots are becoming increasingly popular in assembly processes due to high flexibility and lower cost, hence the development of new optimisation methods suitable for these robots is very desirable. Motivated by previous results showing efficiency of the BA algorithm over other optimisation methods, this research presents a novel adaptation of a hybrid BA discrete algorithms to solve the placement sequence problems in electronic component assembly when using a 6-axis articulated robot for prototype assembly processes. Results of the algorithms are compared with the state of the art GA approach described in [17].

3. Problem Statement

The assembly operation of the printed circuit board can be achieved using variety of placement technologies, however the main focus of this paper is to present the procedure of finding the best solution for a 6-axis robot. Assembly systems typically have a facility for holding the circuit board in place, magazine equipped with feeder racks for component supply and end effectors devices responsible for picking up and placing the components. Once set up the components are sequentially collected from stationary feeders located along the side of the component holding jig and transferred onto their designated place on circuit board. The process optimization will be used to determine an optimal allocation of component feeders and an optimal component placement sequence.

In this paper, we focus on solving the second problem, namely the component placement sequence, while we assume that the component feeder allocation is fixed. Figure 1 shows the example of placement sequence path. Allocation of component feeders and component placement sequence are determined to minimize the total travelled distance by the device responsible for the pick and place operations and hence reduce cycle time.

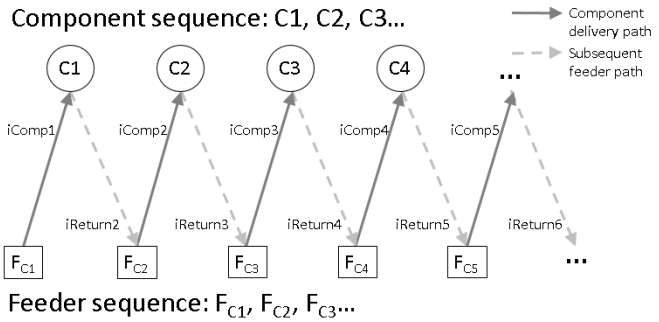

Figure 1. Placement sequence notation

The component sequence problem can be described as the vehicle routing problem with the robot end-effector corresponding to a vehicle with limited loading capacity. The assembly operation works as follows: the end effector traverses to feeder rack, from which it picks up component by using an appropriate tool and travels to place the component on the PCB. After completing the placement tour, the placement head travels back to the magazine, collects another component, and commences the next placement tour [19]. In the placement sequence's problem, the cost function is the sum of all remaining paths, that is the collection of paths from position of each component in sequence to the position of subsequent feeder. In Figure 1 notation, we define it as 'subsequent feeder path', using dashed iReturn vectors. Hence the cost function can be described as:

$$\sum_{i=1}^{n}\sum_{j=1}^{m} D_{ij}\sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2 + \left(z_i - z_j\right)^2} \qquad (3)$$

where:
n – vector describing sequence of component placement destinations
m – vector of consequent feeders from which component are picked
D_{ij} – decision variable, such that:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2021.3082877, IEEE Transactions on Industrial Informatics

$$D_{ij} = \begin{cases} 1 & \text{if following component } j+1 \text{ is stored in feeder } i \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

x_i, y_i, z_i – Cartesian coordinates of the position of *i*th feeder

x_j, y_j, z_j – Cartesian coordinates of the destination of *j*th component

There are however some major assumptions and operation specifications of the assembly system, upon which this investigation is based:

- Each component type is setup only once in the feeder magazine for simplification, as this approach eliminates the component retrieval problem;
- Both the component magazine and the worktable remain stationary and only the robot is mobile;
- The end effector is capable of picking one component at a time.

4. Bat Algorithm for the sequence allocation problem

The BA algorithm's search engine is based on bats searching for prey use sonar based echolocation. By emission of very loud sound pulses and listening for the echo that bounces back from the surrounding objects, the bats are able to navigate in the dark, avoid obstacles, detect and even identify the type of its prey. Their pulses vary in properties and can be correlated with their hunting strategies, depending on the species [26]. Echolocation behaviour was formulated in such way that it can be used as a search algorithm, which in association with the appropriate objective function has proved to be effective way to find optimal solution to many combinatorial problems. Echolocation used by bats is characterized by three parameters: emission rate, loudness and pulse frequency. Emission rate corresponds with the number of pulsed emitted by bat per second and increases as bat reduces distance between it and its prey. The researchers discovered that when bats are hunting, their rate of pulse emission can be increased from 10 up to 200 pulses per second giving the indication of the impressive resolution of bat's sonar and signal processing power.

4.1 Basic Bat Algorithm

The basic algorithm is described in accordance with the following idealized three rules:

1) All bats use echolocation to know the difference between prey and background barriers they navigate within.
2) All bats searching for prey fly randomly with velocity v_i at position x_i with a fixed frequency f_{min}, varying wavelength λ and loudness A_0. The bats are able to adjust the rate and frequency of the emitted pulses.
3) Although the loudness can vary in many ways, on this particular occasion, the assumption is made that the loudness varies from large positive A_0 to a minimum constant A_{min}.

The BA flow chart is presented in Figure 2. First, the virtual bat population is initialized, which involves defining position, velocity, frequency, emission rate and loudness for each of the virtual bats generated. Next, the iteration process begins by generating new solutions through adjusting frequency and updating velocities and positions as shown in equations from 5 to 7 below

$$f_i = f_{min} + (f_{max} - f_{min})\beta, \quad (5)$$

$$v_i^t = v_i^{t-1} + (x_i^t - x_{best})f_i, \quad (6)$$

$$x_i^t = x_i^{t-1} + v_i^t, \quad (7)$$

where β is random number within the range from 0 to 1, f_{min}, f_{max} are minimum and maximum frequency values respectively, x_{best} designates the current global best position found to the point by comparing the values of the cost function of all the solutions. For the local search part, which is subjected to verification of condition: $> r_i$, a new solution is generated by performing a local random walk. A new position (solution) is generated according to equation 6, which states:

$$x_{new} = x_{best} + \varepsilon A^t, \quad (8)$$

where ε is a number in range [-1, 1] and A^t is the average loudness of all the bats within this virtual generation. After this step, condition validation follows such as $rand < A_i$ and $f(x_i) < f(x_{best})$. For each bat that accepts the condition, the emission rate and loudness are updated as follows:

$$r_i^{t+1} = r_i^0[1 - \exp(-\gamma t)], \quad (9)$$

$$A_i^{t+1} = \alpha A_i^t, \quad (10)$$

where γ and α are constants. The algorithm runs until the termination criteria are satisfied.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2021.3082877, IEEE Transactions on Industrial Informatics

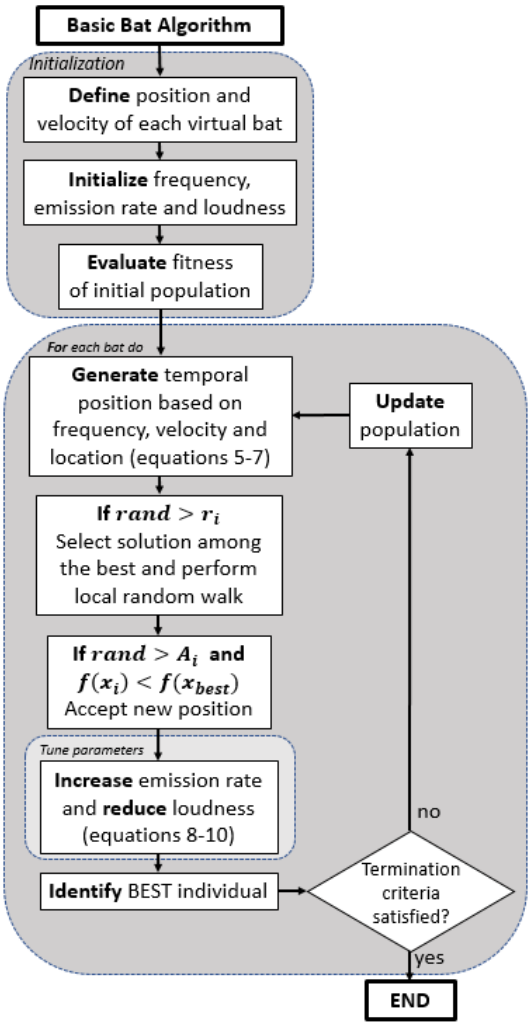

Figure 2. BA flow chart

4.2 Global search: velocity and position updating

In order to shift the BA into discrete space, some level of modification is required. In our proposed discrete algorithm, we have implemented the intelligent link between all the bats in the population. Hence, during the global search, each bat moves in a way depending on its position in relation to the best bat of the population. To represent the solution/position of each bat, we use a n-dimensional vector x_i which is characterized by nodes $(x_{i1}, x_{i2}, \dots, x_{in})$. As for the velocity representation, we used swap sequence procedure, described by $v_i = \{c_{i1}, c_{i2} \dots . c_{im}\}$. This procedure was first presented by Wang et al. [35], and Amara et al. [27] has implemented similar concept in their version of hybrid discrete BA. The swap sequence is an array of swap operators c_i, each including indexes of two nodes that when exchanged in the current solution, allow the response to get closer to the global best solution $x_{best} = (x_{best1}, x_{best2}, \dots, x_{bestn},)$. Hence, the velocity of each bat is the collective of all of the swap operators required to move from current position x_i to global best position x_{best}. Each bat that is going to perform a movement, examines its velocity first. The move the bat performs, depends largely on the frequency. In our approach, the frequency f_i is the probability with which each of the swap operators is to generate the new solution. To improve convergence rate, we added a condition that only the swap operators, which produce an improvement to overall solution quality, are executed. For example, if $f_i = 0.4$ and the swap sequence contains seven swap operators, for each operator there is 4/10 chance that the operator will be applied, assuming that each operator brings an improvement to the solution. Figure 3 presents swap sequence mechanism. As pointed out by Wang et al. [35] the order of swap operators within a swap sequence is significant from the point of view of solution. The application of similar swap operators in distinct order may produce a distinct solution from the original solution. In the example shown, the current position is $x_i^t = (7,2,5,3,1,4,6)$ and the best position is $x_{best} = (4,5,7,6,1,2)$. The velocity is the array of swap operators required to transform the current position into best position, hence: $v_i^t = [(1,6),(2,3),(3,6),(4,7),(6,7)]$. The frequency of the current bat when flying was randomly chosen as 0.4, therefore 40% of the swap operators randomly selected among the group of five will be applied. The greyed area within the array shows that operators (2,3) and (3,6) were selected. The new position is the current position after application of two swap operators according to the description above.

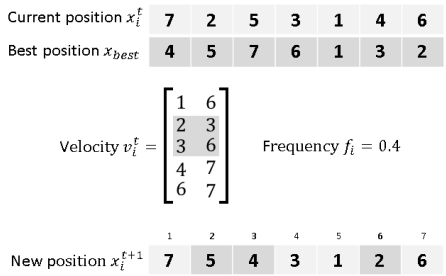

Figure 3. position update based on velocity and frequency

4.3 Parameter tuning

The performance of an algorithm depends largely on the parameters of the algorithm. In BA, parameter control can be done in such a manner that the values of the parameters that include the loudness and rate of pulse emission can be varied as the iterations proceed. In this way, the BA provides inbuilt mechanism to automatically move from the exploration stage to exploitation stage when the optimal solution is approaching [36]. Frequency probability was set up in a way that assist the global search in early stage of the iterations. The probability check (rand>f, where rand is random number in range from 0 to 1 allows high rate of finding new solution in early stages, favouring quick convergence towards good quality solutions. The sigmoid function used to describe the change of frequency as generation progress is presented in equation 11.

$$f_i = A * (1 + \exp(-\beta * (t - \varepsilon))^{-1} \qquad (11)$$

where, A, β and ε are parameters that determine the shape and position of the function.

In our method, the loudness is updated according to equation 10. The additional condition was introduced such that, loudness is reduced only if the new solution sees improvements with

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2021.3082877, IEEE Transactions on Industrial Informatics

regard to previous step, which indicated that those bats are moving towards the optimal position. The emission rate is the force driving the local search. Equation 12 maintains the change of emission rate and a sigmoid function updates the rate as the iteration proceeds.

$$r_i = B + A * (1 + \exp(\beta * (t - \varepsilon))^{-1} \qquad (12)$$

where, A, B, β and ε are parameters that determine the shape and position of the function.

This constructed parameter control introduces an inbuilt capability to automatically transit into areas where there is possibility of obtaining promising solutions by intensifying the local search. This transition or zooming is accompanied by the automatic switching from diversified explorative moves to local intensified exploitative moves, which results in the high convergence rate in early stages of the iterations, compared to other algorithms (Chawla & Duhan 2015).

5. Experimental setup

The performance of the proposed BA to find the best assembly sequence that reduces cycle time is evaluated against a more traditional GA. In the study a KUKA KR16 6 axis industrial robot was used together with component feeders and two PCB components. The following section will describe the robot and the component.

5.1 Eurorack Serge filter as an example product for optimised assembly

The Eurorack Serge filter was selected for this study due to its complexity and variety of components (figure 4). The assembly of a PCB can be achieved with a wide range of technologies. In this research the procedure will be completed with a 6-axis robot. The component provides a good representation of the PCBs that would be prototyped or produced in small batches for beta testing rather than that of an established high-volume system. The assembly starts with placement of the PCB boards in the main fixture, then robot end effector places components collected from fixed feeders located in close proximity to the boards. The Eurorack Serge filter consists of seven variants of 100 components that mainly consists of resistors, capacitors, diodes, ICs and power connectors. Once all of the components are assembled in sequence the parts can be soldered in position.

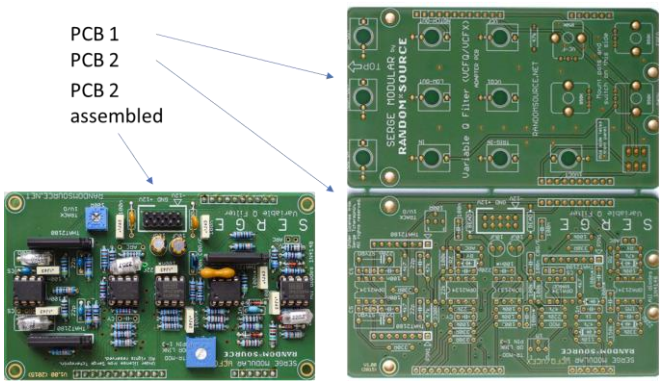

Figure 4. Eurorack Serge Filter PCBs

5.2 KUKA Robot

The study used a KUKA KR16 6 axis industrial robot with component feeders and two PCB components placed in an optimum position within the assembly workspace. All of the assembly positions were programmed using KUKA KRL language (figure 5). To benchmark the algorithms the initial program sequence was based on a lead through (LT) program that started from the top of the main PCB and worked through each position until it reached the bottom, this method was then repeated for the second PCB (figure 6). The GA and BA (figure 7) sequences were then programmed. For each program a timer command was added in order to compare the speed of the program. The KUKA robot has six different types of path motion, and four of these are relevant to the proposed assembly. The motion types are as follows:

- Point to Point (PTP) – This motion type involves following the quickest path between two points and is not necessarily a straight path.
- Linear (LIN) – The linear motion type follows a straight path and uses more joints in constant motion to trace the straight path.
- Spline Point to Point (SPTP) – This is similar to the PTP motion, however it allows for continuous spline motions where points are estimated and a smoother motion is available.
- Spline Linear (SLIN) – As with the SPTP this motion type uses splines between linear motions.

Also, for each of the motions used there are velocity controls (percentage based). The efficacy of the algorithms can be influenced by the various motion types and the speed setting of the robot, therefore motion types and speed controls were applied to each of the programs. By comparing all of the robot potential in terms to motion type and speed it will be possible to identify the relationships between the algorithms and the assembly method.

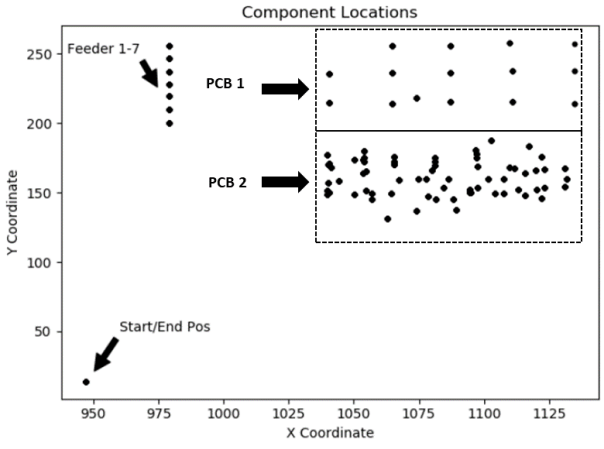

Figure 5. Assembly positions

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2021.3082877, IEEE Transactions on Industrial Informatics

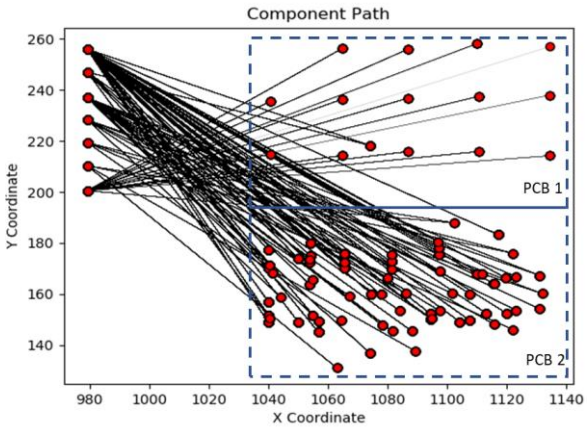

Figure 6. LT sequence

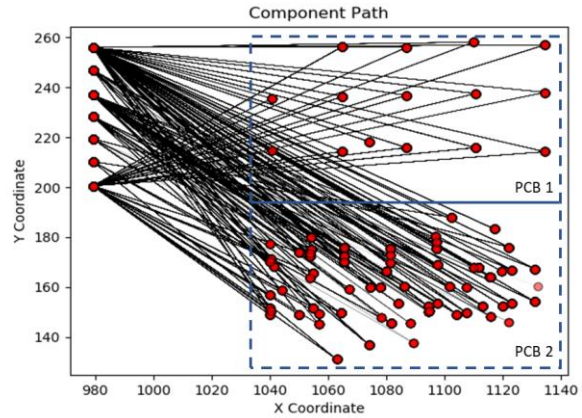

Figure 7. BA sequence.

6. Results

For the Optimal Motion Type and Placement sequences thirty-six practical experiments were used to sequence the placing of individual components on two PCBs, where the main objective was to minimize the total distance travelled by the robot. Using the solutions created with a LT program and the optimized BA and GA, a time difference that also considers robot motion type (LIN, PTP, SLIN, SPTP) and velocity has been found (Table 1). The main results are as follows:

- The BA and GA sequences are faster than LT sequences for all of the motion types and at all velocities tested.
- For all three sequences LIN motions are the slowest and PTP is the fastest (figure 8).
- The BA is marginally better than the GA for LIN, PTP and SLIN, but for the SPTP motion the GA is faster (table 1).
- When comparing the algorithms relationship to the velocity setting of the robot, overall it can be seen that as the robot goes faster the influence of the algorithm is reduced (figure 9).
- The BA when using PTP motion is the sequence that is more influenced by the velocity of the robot.
- Both algorithms when run in SLIN motion are less influenced by the robot velocity.

In terms of efficiencies, table 2 shows significant differences when comparing heuristic sequencing to LT sequencing. The difference between the slowest build is (LT with a LIN motion of 30% velocity) and the quickest (BAT with a PTP at 75% velocity) is 84.26%. If we compare the LT and BAT builds across all velocities, you see an efficiency between 64.54-71.43%. Even for a build of 100 components this is significant and scaling up the production of PCBs with more components will see similar the advantages to the manufacturers

Table 1. LT, BA and GA build times for different robot motions and velocities.

Robot motion type	LIN	PTP	SLIN	SPTP
velocity %	LT build time [ms]			
30	**400404**	114684	243552	246756
50	251952	80208	152892	154800
75	177720	**63084**	107472	108828
	BA build time [ms]			
30	**397116**	114384	242508	245112
50	249972	80052	152208	153768
75	176376	**63012**	107016	108168
	GA build time [ms]			
30	**397188**	114480	242604	244368
50	250008	80124	152268	153408
75	176424	**63048**	107076	107916

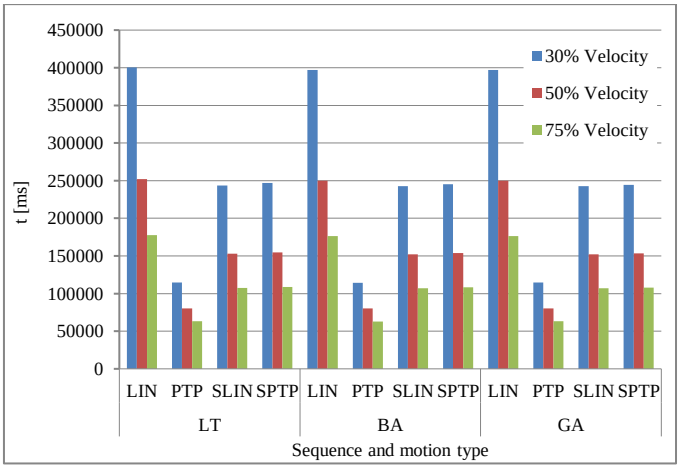

Figure 8. Assembly time for sequence and motion types

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2021.3082877, IEEE Transactions on Industrial Informatics

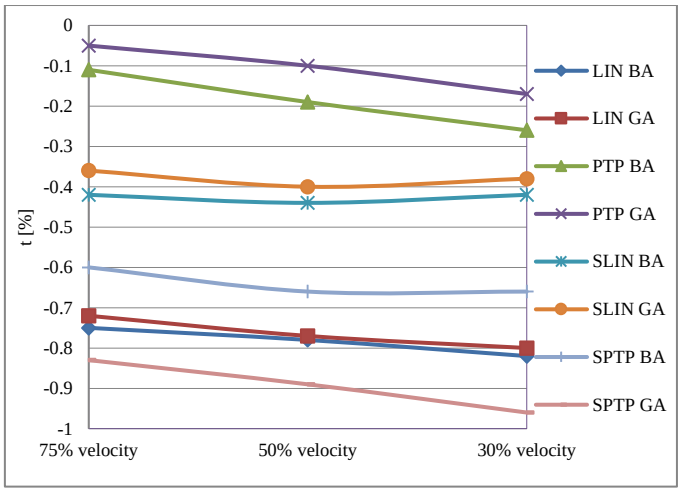

Figure 9. The influence of robot velocity on BA and GA sequences and motion types

Table 2. Efficiency differences

Velocity	slowest	fastest	Optimised efficiency (100 components)
30%	LT LIN 400404 [ms]	BA PTP 114384 [ms]	71.43% increase
50%	LT LIN 251952 [ms]	BA PTP 80052 [ms]	68.22% increase
75%	LT LIN 177720 [ms]	BA PTP 63012 [ms]	64.54% increase
Difference between whole range			
Slowest	Fastest	84.26% increase	
LT LIN 30% velocity 400404 [ms]	BA PTP 75% 63012 [ms]		

7. Conclusions

This research investigated the use of an automation cell that utilised a 6-axis robot to assemble an electronic component. The manufacturing process required an optimal placement sequence for improved production efficiency. To achieve this a novel discrete hybrid BA and a GA was used for solving the feeder slot assignment and placement sequence problem. Numerical experiments were conducted to identify the effectiveness of the algorithm and validation was performed using a KUKA KR16 robotic assembly cell with various path planning programs and motion velocities.

- The BA and GA provide a better solution to reducing the the total distance travelled by the robot when compared to LT programming solutions. A LT sequence using LIN motions at 30% resulted in the slowest assembly time (400404 ms) and Bat sequence using PTP motions at 75% velocity was the fastest (63012 ms). The optimised order sequence reduces the total distance travelled by the KUKA robot and a theoretical time saving of 84.26% could be obtained through optimisation leading to an important improvement in assembly time.
- Experimental results were tested using the KUKA robot to validate the algorithms ability to improve placement sequences. Knowing that four different robot motion types are used in real automation sequences, each was compared. For the LT and BA and GA sequences LIN motions are the slowest and PTP is the fastest. The BA is marginally better than the Genetic algorithm for LIN, PTP and SLIN, but for the SPTP motion the GA is faster.
- In addition to automation movement types, three different velocities were considered for each sequence type and each movement type. The research shows that there is a relationship between the algorithms and the velocity setting of the robot. As the robot goes faster the influence of the algorithm is reduced. The BA when using PTP motion is the sequence that is more influenced by the velocity of the robot. Both BA and GA sequences are less influenced by the robot velocity when using SLIN motions. Importantly it should be noted that in larger more complex programs with more complex part geometries it is likely that the programmer will used multiple speeds and motion types to meet the specifications of the assembly.
- 6 axis robots provide an automation solution that allows for flexible manufacturing. The manipulation capabilities they possess mean that they can be used for traditional activities but also complete complex tasks that are normally associated with specialised automation. This research has shown that a BA inspired and GA can be used with 6 axis robots to optimise PCB assembly. With a focus on makespan minimization, the algorithms out performed traditional LT programs. However, the results also showed that when using such a sequence optimization method the programmer must holistically consider the different robot motion types and the velocities in which the robot operates with.
- In future studies the findings of this research will be focused in two directions. The first is to modify the algorithm to use multiple robot motions within a single program. A rule based system will also be used to recognise the accuracies achievable with different velocities. The second direction will be to verify the accuracy of the robot interface when commanded to modify a trajectory and re sequence an established path.

References

[1] Castellani, M., Otri, S., & Pham, D. T. (2019a). Printed circuit board assembly time minimisation using a novel Bees Algorithm. *Computers and Industrial Engineering*, *133*(April), 186–194. https://doi.org/10.1016/j.cie.2019.05.015

[2] Crama, Y., Van De Klundert, J., & Spieksma, F. C. R. (2002). Production planning problems in printed circuit board assembly. *Discrete Applied Mathematics*, *123*(1–3), 339–361. https://doi.org/10.1016/S0166-218X(01)00345-6

[3] Nilakantan, J. M., Huang, G. Q., & Ponnambalam, S. G. (2015). An investigation on minimizing cycle time and total energy consumption in robotic assembly line systems. *Journal of Cleaner Production*, *90*, 311–325. https://doi.org/10.1016/j.jclepro.2014.11.041

[4] Altinkemer, K., Kazaz, B., Köksalan, M., & Moskowitz, H. (2000). Optimization of printed circuit board manufacturing: Integrated modeling and algorithms. *European Journal of Operational Research*, *124*(2), 409–421. https://doi.org/10.1016/S0377-2217(99)00169-1

[5] Ji, P. (2008). Minimizing Cycle Time for PCB Assembly Lines : An Integer Programming Model and a Branch-and-Bound Approach. *Information & Management Sciences*, *19*(2), 237–243.

[6] Kodek, D. M., & Krisper, M. (2004). Optimal algorithm for minimizing production cycle time of a printed circuit board assembly line. *International Journal of Production Research*, *42*(23),50315048.https://doi.org/10.1080/00207540412331285814 *40*(4), 293–307. https://doi.org/10.1016/S0360-8352(01)00030-4

[7] Ong, N., & Tan, W. (2002). Sequence placement planning for high-speed PCB assembly machine. *Integrated Manufacturing Systems*, *13*(1), 35–46. https://doi.org/10.1108/09576060210411495

[8] Ball, M. O., & Magazine, M. J. (1988). Sequencing of insertions in printed circuit board assembly. *Operations Research*, *36*(2), 192–201. https://doi.org/10.1287/opre.36.2.192

[9] Ahmadi, R. H., & Mamer, J. W. (1999). Routing heuristics for automated pick and place machines. *European Journal of Operational Research*, *117*(3), 533–552. https://doi.org/10.1016/S0377-2217(98)00231-8

[10] Ho, W., & Ji, P. (2005). A genetic algorithm to optimise the component placement process in PCB assembly. *International Journal of Advanced Manufacturing Technology*, *26*(11–12), 1397–1401. https://doi.org/10.1007/s00170-004-2132-5

[11] Loh, T. S., Bukkapatnam, S. T. S., Medeiros, D., & Kwon, H. (2001). A genetic algorithm for sequential part assignment for PCB assembly. *Computers and Industrial Engineering*, 40,2001,293-307.

[12] Ong, N. S., & Khoo, L. P. (1999). Genetic algorithm approach in PCB assembly. *Integrated Manufacturing Systems*, *10*(5), 256–265. https://doi.org/10.1108/09576069910280648

[13] Wang, W. P., & Tseng, H. E. (2009). Complexity estimation for genetic assembly sequence planning. *Journal of the Chinese Institute of Industrial Engineers*, *26*(1), 44–52. https://doi.org/10.1080/10170660909509120

[14] Broad, K., Mason, A., Rönnqvist, M., Frater, M., Broad, K., Mason, A., … Frater, M. (1996). rch Society Stable URL : Optimal Robotic Component Placement. *The Journal of the Operational Research Society*, *47*(11), 1343–1354.

[15] Chen, Y. M., & Lin, C. T. (2007). A particle swarm optimization approach to optimize component placement in printed circuit board assembly. *International Journal of Advanced Manufacturing Technology*, *35*(5–6), 610–620. https://doi.org/10.1007/s00170-006-0777-y

[16] Alkaya, A. F., & Duman, E. (2015). Combining and solving sequence dependent traveling salesman and quadratic assignment problems in PCB assembly. *Discrete Applied Mathematics*, *192*(2015), 2–16. https://doi.org/10.1016/j.dam.2015.03.009

[17] Andrzejewski, K. T., Cooper, M. P., Griffiths, C. A., & Giannetti, C. (2018). Optimisation process for robotic assembly of electronic components. *International Journal of Advanced Manufacturing Technology*, *99*(9–12), 2523–2535. https://doi.org/10.1007/s00170-018-2645-y

[18] Moghaddam, M., & Nof, S. Y. (2016). Parallelism of Pick-and-Place operations by multi-gripper robotic arms. *Robotics and Computer-Integrated Manufacturing*, *42*, 135–146. https://doi.org/10.1016/j.rcim.2016.06.004

[19] Grunow, M., Günther, H. O., Schleusener, M., & Yilmaz, I. O. (2004). Operations planning for collect-and-place machines in PCB assembly. *Computers and Industrial Engineering*, *47*(4), 409–429. https://doi.org/10.1016/j.cie.2004.09.007

[20] Ellis, K. P., Vittes, F. J., & Kobza, J. E. (2001). Optimizing the performance of a surface mount placement machine. *IEEE Transactions on Electronics Packaging Manufacturing*, *24*(3), 160–170. https://doi.org/10.1109/6104.956801

[21] Deo, S., Javadpour, R., & Knapp, G. M. (2002). Multiple setup PCB assembly planning using genetic algorithms. *Computers and Industrial Engineering*, *42*(1), 1–16. https://doi.org/10.1016/S0360-8352(01)00062-6

[22] Ho, W., & Ji, P. (2005). A genetic algorithm to optimise the component placement process in PCB assembly. *International Journal of Advanced Manufacturing Technology*, *26*(11–12), 1397–1401. https://doi.org/10.1007/s00170-004-2132-5

[23] Kulak, O., Yilmaz, I. O., & Günther, H.-O. (2007). PCB assembly scheduling for collect-and-place machines using genetic algorithms. *International Journal of Production Research*, *45*(17), 3949–3969. https://doi.org/10.1080/00207540600791608

[24] Li, S., Hu, C., & Tian, F. (2008). Enhancing optimal feeder assignment of the multi-head surface mounting machine using genetic algorithms. *Applied Soft Computing Journal*, *8*(1), 522–529. https://doi.org/10.1016/j.asoc.2007.02.012

[25] Castellani, M., Otri, S., & Pham, D. T. (2019b). Printed circuit board assembly time minimisation using a novel Bees Algorithm. *Computers and Industrial Engineering*, *133*(May), 186–194. https://doi.org/10.1016/j.cie.2019.05.015

[26] Yang, X. S. (2010). A new metaheuristic Bat-inspired Algorithm. *Studies in Computational Intelligence*, *284*, 65–74. https://doi.org/10.1007/978-3-642-12538-6_6

[27] Amara, J., Hamdani, T. M., & Alimi, A. M. (2015). A new Hybrid Discrete Bat Algorithm for Traveling Salesman Problem using ordered crossover and 3-Opt operators for Bat's local search. In *2015 15th International Conference on*

Intelligent Systems Design and Applications (ISDA) (pp. 154–159). https://doi.org/10.1109/ISDA.2015.7489217

[28] Kaveh, A., & Zakian, P. (2014). Enhanced bat algorithm for optimal design of skeletal structures. *Asian Journal of Civil Engineering*, *15*(2), 179–212.

[29] Osaba, E., Yang, X. S., Diaz, F., Lopez-Garcia, P., & Carballedo, R. (2016). An improved discrete bat algorithm for symmetric and asymmetric Traveling Salesman Problems. *Engineering Applications of Artificial Intelligence*, *48*, 59–71. https://doi.org/10.1016/j.engappai.2015.10.006

[30] Saji, Y., & Riffi, M. E. (2016). A novel discrete bat algorithm for solving the travelling salesman problem. *Neural Computing and Applications*, *27*(7), 1853–1866. https://doi.org/10.1007/s00521-015-1978-9

[31] Musikapun, P., & Pongcharoen, P. (2012). Solving Multi-Stage Multi-Machine Multi-Product Scheduling Problem Using Bat Algorithm. *International Conference on Management and Artificial Intelligence*, *35*, 98–102.

[32] Asma, C., Rabia, K., Mohamed, B and Xin-She, Y.(2017) New directional bat algorithm for continuous optimization problems *Expert Systems With Applications 69 159–175.*

[33] Lijue, L., Shuning, L., Fan, G and Shiyang, T. (2020) Multi-point shortest path planning based on an Improved Discrete Bat Algorithm. *Applied Soft Computing Journal 95 (2020) 106498*

[34] Saraswathi M., Bala G., Murali, B., Deepak, BBVL (2018). Optimal Path Planning of Mobile Robot Using Hybrid Cuckoo Search-Bat Algorithm. *Procedia Computer Science 133 (2018) 510–517*

[35] Wang K.-P., Huang L., Zhou C.-G., and Pang W. (2003), "Particle swarm optimization for traveling salesman problem," in Machine Learning and Cybernetics, 2003 *International Conference on, vol. 3. IEEE*, pp. 1583-1585.

[36] Chawla M. and Duhan M. (2015), "Bat Algorithm: A survey of the state-of-the-art", *Applied Artificial Intelligence*, 29: 617–634.