The importance of being constrained: dealing with infeasible solutions in Differential Evolution and beyond

Anna V. Kononova a.kononova@liacs.leidenuniv.nl
LIACS, Leiden University, The Netherlands

Diederick Vermetten d.l.vermetten@liacs.leidenuniv.nl
LIACS, Leiden University, The Netherlands

Fabio Caraffini fabio.caraffini@swansea.ac.uk
Department of Computer Science, Swansea University, United Kingdom

Madalina-A. Mitran madalina.mitran96@e-uvt.ro
Department of Computer Science, West University of Timişoara, Romania

Daniela Zaharie daniela.zaharie@e-uvt.ro
Department of Computer Science, West University of Timişoara, Romania

Abstract

We argue that results produced by a heuristic optimisation algorithm *cannot* be considered reproducible unless the algorithm fully specifies what should be done with solutions generated outside the domain, *even* in the case of simple bound constraints. Currently, in the field of heuristic optimisation, such specification is rarely mentioned or investigated due to the assumed triviality or insignificance of this question. Here, we demonstrate that, at least in algorithms based on Differential Evolution, this choice induces notably different behaviours in terms of performance, disruptiveness and population diversity. This is shown theoretically (where possible) for standard Differential Evolution in the absence of selection pressure and experimentally for the standard and state-of-the-art Differential Evolution variants, on a special test function and the BBOB benchmarking suite, respectively. Moreover, we demonstrate that the importance of this choice quickly grows with problem's dimensionality. Differential Evolution is *not* at all special in this regard – there is no reason to presume that other heuristic optimisers are not equally affected by the aforementioned algorithmic choice. Thus, we urge the heuristic optimisation community to formalise and adopt the idea of a *new algorithmic component* in heuristic optimisers, which we refer to as the strategy of dealing with infeasible solutions. This component needs to be consistently: (a) specified in algorithmic descriptions to guarantee reproducibility of results, (b) studied to better understand its impact on an algorithm's performance in a wider sense (i.e. convergence time, robustness, etc.) and (c) included in the (automatic) design of algorithms. All of these should be done *even* for problems with bound constraints.

Keywords

bound constraint, algorithmic behaviour

1 Introduction

The overwhelming majority of practical optimisation problems are constrained at least in some sense: from simply limiting the ranges of input variables to complex nonlinear or black-box functional constraints. There is a clear benefit of tackling such problems according to their constrained nature by considering the search space as constrained rather than at first approximating the problem as an unconstrained variant.

It is true that both engineering and pure mathematical approaches dictate that infeasible solutions should not be evaluated during optimisation: the former - due to physical limitations of the underlying processes/devices, and the latter - since the objective function is not required to be formally defined for such infeasible solutions. However, heuristic optimisation approaches, not being exact, sometimes use some kind of 'information' on the values of the objective function corresponding to infeasible solutions – see exterior penalty approaches (Coello Coello, 2002). On one hand, this narrows down the applicability of such methods. On the other hand, they get an advantage through additional domain information. Is it fair to compare such algorithms with those not using this information? In our view, such distinction should *at least be highlighted*. Additionally, when benchmarking heuristic optimisation algorithms, it is often unclear whether the boundaries should be dealt with explicitly by the algorithm, or if the problem itself should handle this using a penalty function (Hansen et al., 2021).

While the current level of development of the field of numerical heuristic optimisation allows tackling these problems directly as constrained, it is not done consistently, especially for the simplest types of constraints. For example, very few papers in the field mention what should be done with infeasible solutions (that violate some constraint) that might, or even are very likely to, be generated during an optimisation run. Many options are possible for handling such solutions, e.g. penalty functions, repair methods, and feasibility-preserving generating operators. In practice, the choice made for a particular algorithm is often omitted from the description due to either its assumed insignificance or triviality in the eyes of the algorithm's designer. However, this choice has been shown to strongly influence the algorithm's performance (Arabas et al., 2010; Liao et al., 2014; Biedrzycki et al., 2019; Boks et al., 2021; Kononova et al., 2021).

Moreover, the aforementioned ambiguity in algorithms' specifications regarding dealing with infeasible solutions naturally leads to *reproducibility issues*. It is this aspect specifically that is being discussed in this paper.

If the algorithm's source code is not made available, such an ambiguity in algorithms' specification has to be resolved via wasteful trying ('and-erroring') of infinite possibilities. The latter fact is further burdened by the random nature of the algorithms which does not allow guesses regarding missing details exactly: unless, in addition to the available source code, the *exact* specifications of the operating system where the reported experiments have been executed, versions of programming languages and random seed values of pseudorandom number generator are also known (L'Ecuyer and Simard, 2007; van den Honert et al., 2021).

If the source code of an algorithm is available, it is not guaranteed that missing algorithmic specification can be easily extracted and therefore the accompanying specification can still be ambiguous regarding the strategy for dealing with infeasible solutions. We can assume that either multiple options have been tried and the selected one performs best, or the strategy for dealing with infeasible solutions component was considered as non-essential to the algorithm behaviour and thus, not investigated. In either case, information about the algorithm in relation to the strategy of dealing with infeasible solutions is never reported, and when only one method was considered this could mean that *potential performance gains* have not been realised. This same argument can hold in the cases of automatic algorithm design and configuration, where modules dealing with the bound constraints are rarely part of the search-space. Even when they are considered in the configuration, they are often grouped together with another operator such as mutation, thus making it more challenging to accurately see its impact (Stützle and López-Ibáñez, 2019; Cruz-Duarte et al., 2020).

While the wider community of evolutionary algorithms has recently become more aware of the challenges and benefits of reproducibility (López-Ibáñez et al., 2021), the standards for availability of code, data, and other artefacts still differ widely between the venues. For the field of heuristic optimisation algorithms to move forward, a stronger reliance on reproducible experimental results is needed, especially in the absence of overarching theoretical frameworks. Thus, in order to ensure that any reported findings can be reproduced, we should aim to be aware of even the seemingly small design decisions within our algorithms that are often overlooked, since even minor changes in algorithm search behaviour can lead to irreproducible results if not properly documented or otherwise made available.

Unfortunately, many of the papers which propose new (meta)heuristics or algorithmic improvements on existing ones do not contain explicit information on how the out-of-bounds components are treated. In the particular case of Differential Evolution (DE), the number of such papers is significant – see Section 2.5 for a literature review. We believe such problem manifests itself for *all heuristic optimisation methods* discussed in both specialised theoretical and applied literature. Thus, with this paper, we conclude that there is a *major reproducibility issue* with the state-of-the-art heuristic optimisation methods and call for proper formalisation of a new operator/algorithmic component that deals with infeasible solutions inside heuristic optimisers. Such component needs to be *consistently*:

(a) specified in algorithmic descriptions to guarantee full reproducibility of results,

(b) studied to understand its impact on algorithms' performance in a wider sense (i.e.

convergence time, robustness, etc.),

(c) included in the (automatic) design of algorithms.

All of the above should be done *even* for problems with bound constraints only.

To emphasise the importance of such an algorithmic component, we propose an integrated approach, at both experimental and theoretical levels, to analyse the impact of the strategies used to deal with solutions violating bound constraints on the behaviour of the optimisation algorithm. One of the main contributions of this paper is the usage of the 'cosine similarity' measure to quantify the influence of the aforementioned strategies on the search direction induced by the optimisation algorithm.

The remainder of this paper is organised as follows: In Section 2, we discuss the general problem of using heuristic optimisation methods to solve bound-constrained problems, with a focus on Differential Evolution and the Strategy of Dealing with Infeasible Solutions (SDIS) used in this context. In Section 3, we consider the notion of disruption of the search behaviour and propose the usage of cosine similarity to measure this phenomenon. Then, in Section 4, we present a theoretical analysis on the number of infeasible solutions within DE, and analyse the impact of several popular SDIS on search directions and diversity of the population. To further analyse the impact of SDIS on these aspects, we make use of the special test function f_0, which assigns uniformly distributed random values to the elements and hence 'removes' the selective pressure *without* any modification to the algorithm under investigation, and uses this to study the relation between parameters of DE, SDIS, cosine similarity between search directions and population diversity. We also consider the overall number of infeasible solutions generated, and relate the analysis on f_0 to the concept of structural bias. Finally, we perform a benchmark study on several versions of DE and investigate the empirical impact of SDIS on their performance, while comparing the algorithmic behaviour observed to the results obtained theoretically on flat functions and empirically on f_0. In Section 7, we conclude that SDIS does indeed have an impact on algorithms' performance, and should be taken into consideration more closely to improve the state of reproducibility in our field. We look ahead at potential solutions and future research directions in Section 8.

2 Heuristic optimisation with bound constraints

Optimisation problems faced by practitioners from different application fields are necessarily defined within a domain $\mathbf{D}$, commonly referred to as the search space in the heuristic optimisation community. Indeed, in the real-world context, the presence of feasibility constraints is almost inevitable, and even when the nature of the problem seems to be unconstrained one may argue that when using heuristic approaches the need for sampling solutions and generating random numbers imposes boundaries for drawing such solutions. In this light, even when explicit equality and/or inequality constraints are not present, it is generally assumed that each design variable of the problem at hand must be bounded between some lower and upper bound, thus defining a search space $\mathbf{D}$ shaped as a hyperparallelepiped. Such a setting is commonly referred to as bound-constrained or 'box-constrained' problem in computer science jargon. In this study, we focus on *real-valued single objective bound-constrained optimisation problems*, as defined and discussed in the next Section.

2.1 Problem formulation, related constrained optimisation problems

A *real-valued bound-constrained problem* is defined as finding a minimum of function

$$f : \mathbf{D} = \bigtimes_{i=1}^{n} [a_i, b_i] \to \mathbb{R} \tag{1}$$

where $-\infty < a_i < b_i < \infty$ and $\mathbf{D} \subset \mathbb{R}^n$ [1]. This represents the lowest complexity of inequality constraint condition on the variables that a problem can have. For this reason,

[1]If $X \subset \mathbb{R}^n$ defines a bound-constrained domain, any function $f : X \to Y \subset \mathbb{R}$ can be extended to a larger domain as $g : \mathbb{R}^n \to W \subset \mathbb{R}$ (i.e. $g(x) = f(x)\ \forall x \in X$) and original bound-constrained problem can be rewritten as an *optimisation problem with unbounded domain subject to inequality constraint*:

$$\arg\min_{x \in \mathbb{R}^n} g(x) \text{ subject to } v(x) \leqslant 0,$$

where $v(x) = \mathbf{A}\mathbf{x} - \mathbf{b}$ with $\mathbf{A} = [-\mathbf{I_n}, \mathbf{I_n}]^\mathbf{T}$ and $b = [-a_1, \ldots, -a_n, b_1, \ldots, b_n]^\mathbf{T}$, $v : \mathbb{R}^n \to \mathbb{R}^{2n}$, where $\mathbf{I_n}$ stands a unity matrix of size n. In other words, trivially, bound constraints represent a special case of a set of linear constraints. However, *in practice*, such application of the latter methods to the transformed original problem might lead to poor results, e.g. depending on the way the function at hand is extended (from f to g, in the notation above), since an infinite number of extensions can be defined for any function – see above the ambiguity in defining values of $g(x) \forall x \notin X$.

some confusion arises in the literature with several authors often referring to this class of problems as 'unconstrained' to stress the fact that design variables are not subject to more complex linear or nonlinear constraints. However, we argue that this is incorrect because ignoring bound constraints is an oversimplification leading to confusion and reproducibility issues. It is indeed common to find articles in the literature where information on the employed Strategy of Dealing with Infeasible Solutions (SDIS, see definition in Section 2.4) is omitted (see Section 2.5), even though recent studies indicate that different SDIS operators differently influence (at least) such a characteristic of a heuristic optimisation method as structural bias (see Section 5.5 for definition) (Kononova et al., 2015; van Stein et al., 2021; Vermetten et al., 2022c,d), and thus playing a role on the algorithmic behaviour of an optimisation algorithm. Hence, bound constraints *should not* be ignored and solutions violating them are to be dealt with an appropriate SDIS. In this light, a well-designed algorithm for real-valued unconstrained optimisation, i.e. where each design variable can be anywhere in the real axis ($x_i \in \mathbb{R}$), might not be as suitable for bound-constrained optimisation. It should indeed be observed how some algorithms, see e.g. (Helwig and Wanka, 2008; Engelbrecht, 2013a; Kononova et al., 2021), are prone to produce high numbers of solutions outside the search domain under certain parameter configurations. This is quite likely to occur when an algorithm for unconstrained optimisation is used over a bound-constrained domain, particularly when the optimum is close to the boundary of the domain. In this scenario, the algorithm has to be equipped with an SDIS which would have to be activated for the vast majority of objective function evaluations, thus leading the search and taking over the actual working logic of the algorithm itself. To prevent this phenomenon, being *constrained* would be a key feature of an optimisation algorithm for bound-constrained problems.

2.2 Classic and state-of-the-art versions of Differential Evolution

Despite the numerous advances in the field of DE, its solid general algorithmic framework has remained quite unchanged since the first studies (Storn, 1996; Storn and Price, 1997), with many of the most important variants being proposed by mainly acting on the mutation operator, where individuals are linearly combined, see (Lampinen and Zelinka, 2000; Price et al., 2006), and on adding self-adaptation rules for its three parameters (Das et al., 2016). These are the population size N and the two control parameters $F \in (0, 2]$, acting as a *scale factor* for the mutation operator, and the *crossover rate* $C_r \in [0, 1]$. The working mechanism of DE is quite known and established, and for general information one can see (Price et al., 2006; Das et al., 2016; Opara and Arabas, 2019; Kononova et al., 2021), where description, pseudocode and analyses of its algorithmic behaviour are provided. However, for the sake of clarity, we briefly report relevant DE terminology which is used in the remainder of this paper.

In DE, the N individuals in the population are processed one at a time. When the commonly called 'current' individual to be perturbed is selected to undergo recombination, it gets referred to as the `target`. Through the crossover operator, which requires the availability of an 'intermediate' solution referred to as the `mutant`, the `target` individual produces an offspring solution referred to as the `trial`. This new solution can have infeasible components to be dealt with an appropriate SDIS before its fitness value can be computed, as further commented in Section 2.4. To implement this logic, a mutation strategy is required to produce the `mutant` solution. As previously mentioned, this operator works by linearly combining individuals selected from the population where a number of difference vectors are formed (from which the name of this optimisation paradigm) and added to a specific individual. The latter, as well as the number of difference vectors, depends on the adopted mutation strategy. With this in mind, classic DE variants are identified with the well-known notation `DE/a/b/c` where `a` indicates the mutation strategy, `b` the number of difference vectors employed in the mutation strategy, and `c` specifies the crossover strategy - two options are mainly used (i.e. the binomial `bin` and exponential `exp` crossover strategies) for `c` but a few more strategies also exist in the literature (Das et al., 2016; Vermetten et al., 2022d).

Some state-of-the-art DE algorithms, which we also study in this piece of research, slightly deviate from this structure. This is the case of the Success-History based Adaptive DE (`SHADE`) (Tanabe and Fukunaga, 2013), which can be seen as a variant of the popular `JADE` algorithm (Zhang and Sanderson, 2009). `SHADE` features a memory system where both the weighted Lehmer average of successful F values, and the weighted arithmetic average of successful C_r values from previous generations are stored. Such values are randomly picked to adapt the control parameters, thus not relying only on the values from the previous generation (as in JADE) but also on the previous ones. Furthermore, the p parameter for the 'current-to-*p*best' mutation strategy is randomly generated for each individual (this introduces an extra parameter p_{min} to tune). These small changes led to reported significant performance improvements with respect to previously established self-adaptive DE algorithms. When compared to the algorithmic structure of a classic DE, one can immediately see a clear difference for SHADE:

- control parameters are self-adapted;
- by design, a mechanism is in place for using an optional archive of less fit individuals to be entered in the population for preserving diversity;
- classic DE mutations are not employed, in favour of the 'current-to-*p*best/1' (Zhang and Sanderson, 2009), where the *p*best vector is selected at random amongst the $p\%$ best individuals in the population, which is used in combination with the binomial crossover in (Tanabe and Fukunaga, 2013).

The burden of tuning the population size is still present in SHADE, but is mitigated in its successor L-SHADE (Tanabe and Fukunaga, 2014), where an initial (usually large, i.e. $18 \cdot n$) population size gets decreased linearly as a function of the number of fitness evaluations. Reduction of the population size has shown to be beneficial in DE, see e.g. (Zamuda and Brest, 2012), and appears to make L-SHADE perform better in several benchmark problems.

2.3 Infeasibility

Referring to Eq. 1, a solution $\mathbf{x} \in \mathbf{D}$ is said to be *feasible*, while it is *infeasible* if $\mathbf{x} \notin \mathbf{D}$. In a bound-constrained scenario, the last case occurs if at least one of its ith design variables is either lower than a_i or greater than b_i. Such *infeasible* solutions cannot be evaluated in the vast majority of real-world applications, i.e. they represent physically impossible scenarios or require mathematically undefined calculations, and are purposely excluded by design. Also from the mathematical point of view, these solutions should not be considered because the function modelling the problem is generally undefined outside its domain - i.e. the problem does not exist outside $\mathbf{D}$. Despite this, some confusion can arise while using common benchmark suites for optimisation such as e.g. (Hansen et al., 2021; Wu et al., 2017), which always return a value for $\mathbf{x} \notin \mathbf{D}$, these solutions should not be used to guide the search for solving test-bed problems.

It should be mentioned that strategies that treat the search as unconstrained might be beneficial, as long as the objective function is well-defined outside the search domain. The impact of 'roaming behaviour' of unconstrained elements has been previously analysed for Particle Swarm Optimisation (Engelbrecht, 2013b) and Differential Evolution (Engelbrecht, 2013a). Despite their interesting behaviour, such strategies are out of the scope of this paper, as we are focusing on the case when the objective function cannot be evaluated for infeasible solutions.

The number of infeasible solutions generated during the search depends both on the particularities of the problem (e.g. fitness landscape, problem size) and on the characteristics of the search heuristic. More specifically, the number of infeasible solutions increases with the problem size, n, and with the probability p_v of violating the bound constraints by a design variable, as the probability of generating an n-dimensional infeasible solution is $1-(1-p_v)^n$. The bound violation probability, p_v, depends on the distribution of the population elements in the bounding box and on the exact mutation or perturbation operator. Theoretical results on the estimation of p_v are scarce, as the distribution of the population can be analytically derived only in the initial stage of the evolution. One of the few papers addressing this aspect is (Helwig and Wanka, 2008), which proves that in the case of Particle Swarm Optimisation, many particles leave the bounding box in the first generation, particularly in the case of high-dimensional problems.

The question whether a well-performing algorithm should generate many infeasible points to solve the problem remains open: (Boks et al., 2021) has demonstrated that highly competitive adaptive variants of the Differential Evolution algorithm (see Section 2.2) can indeed generate excessively large number of infeasible points throughout runs. Similar results have been obtained in this paper particularly on functions with optimum near the boundary (see Figure 9). With these results in mind, can we still claim that such optimisation methods *efficiently utilise information contained within the population* if so many generated solutions need to be brought back into feasibility?

2.4 Strategy of dealing with infeasible solutions

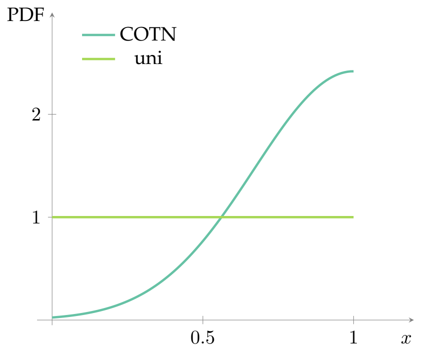
Following the discussions of Section 2.1, SDIS, also referred to as boundary constraint handling methods, are key operators for most algorithms. The same way variation and recombination operators are carefully selected and combined during the algorithmic design phase, SDIS should be considered carefully. The most logical, and most commonly recommended, moment for the use of SDIS in an algorithm is immediately before the objective function call, to make sure that only feasible solutions are evaluated.

While placing the SDIS immediately before evaluation ensures that only feasible solutions are evaluated, there might be some cases, e.g. some hybrid heuristic structures, where intermediary solutions are involved in driving the search process before a new individual is evaluated. Such intermediary solutions should not be used without checking if they are feasible. So, in the most general case, one should always pay attention in activating SDIS every time a potentially infeasible solution is used to guide the search, or has to be evaluated. As suggested in Section 2.2, in the DE framework there is only one operator that can produce an infeasible solution. After being generated, some of its components are transferred by a crossover operator to an existing individual, whose fitness value must be evaluated. Hence, placing a SDIS before crossover would also make sure that novel candidate solutions are feasible, but this might be unnecessary because the crossover could completely ignore infeasible components from the `mutant`. In DE, the `mutant` is never evaluated nor used to guide other parts of the search, so the placement of SDIS before or after crossover are equivalent. In this study, we place SDIS after crossover to match the scheme depicted in the pseudocode from (Kononova et al., 2021).

For this study, we select a varied range of existing strategies for dealing with infeasible solutions:

- 'complete one-sided truncated normal', first introduced in (Caraffini et al., 2019), which is denoted as `COTN` in the remainder of this investigation;
- 'halfway-to-violated-bounds', denoted as `HVB` here, which we define as the operator replacing infeasible components with the midpoint between the previous feasible components (before perturbation) and the violated problem's bound;
- 'mirror', as described in (Kononova et al., 2020b,a), which we denote as `mir` here;
- 'saturation', see (Caraffini et al., 2019), which is denoted here as `sat`;
- 'toroidal', see (Caraffini et al., 2019), which is denoted as `tor` here;
- 'uniform', as defined in (Vermetten et al., 2022d), which we referred to as `uni` here.

Graphically, these SDIS are explained in Figure 1, which brings to attention the stochastic nature of `COTN` and `uni`, while all the remaining strategies deterministically return the same feasible value when the same infeasible value is used as input. Similarly, one can also observe that the way `COTN` operates resembles a stochastic counterpart of `mir`. These selected SDIS operators cover multiple and commonly used working



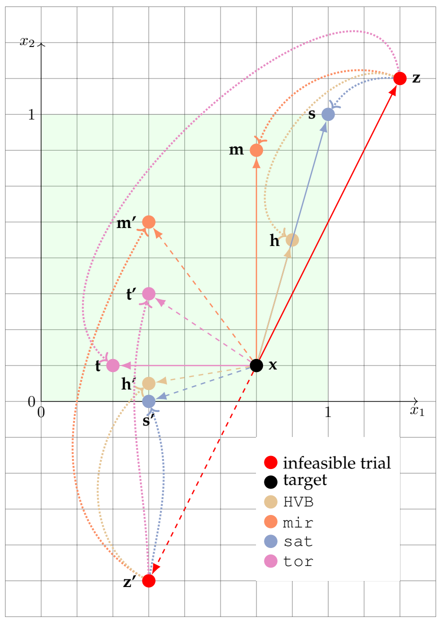


Figure 1: (a) Sampling distributions for the application of stochastic per-component SDIS COTN and uni in case of bound violation on the right for the 1-dimensional unit domain; for bound violation on the left, COTN curve will be symmetric with respect to $x = 0.5$, while uni curve will be the same. (b) Effect of the application of deterministic per-component SDIS HVB, mir, sat, tor (dotted lines) in case of one (dashed lines) and two (solid lines) infeasible components of a trial vector generated from a target vector for 2-dimensional unit hypercube domain (green area).

Table 1: Previous studies on the influence of SDIS on DE, with dictionary for *alternative terminology of SDIS methods*

Paper	Arabas et al. (2010)	Padhye et al. (2015)	Kreischer et al. (2017)	Biedrzycki et al. (2019)	(de-la-Cruz-Martínez and Mezura-Montes, 2020)	Boks et al. (2021)	This paper
Benchmark	CEC 2005	4 functions[2]	CEC 2017	CEC 2017	real-world	BBOB	BBOB
Standard DE variants, (F, C_r)	DE/rand/1/bin, $(0.8, 0.9)$	DE/best/1/bin, $(0.7, 0.5), (0.8, 0.9)$[3]	DE/target-to-best/1/bin, $(0.8, 0.9)$	DE/rand/1/bin, $(0.9, 1)$, DE/local-to-best/1/bin	DE/rand/1/bin, $(\mathcal{U}[0.3, 0.9], \mathcal{U}[0.8, 1])$	15 mutation × 2 crossover operators, SHADE adaptation	DE/rand/1/ with 2 crossovers, 25-50 uniformly spaced (F, C_r) values
Advanced DE variants	–	–	–	SADE, JADE, jSO, DES, BBDE	–	–	SHADE, L-SHADE
Coordinate-wise SDIS	reflection	–	reflection	reflection	reflection	reflection	mirror (`mir`)
	projection	set on boundary	projection	projection	projection	projection	saturation (`sat`)
	wrapping	periodic	–	wrapping	–	wrapping	toroidal (`tor`)
	reinitialisation	random	reinitialisation	reinitialisation	random	reinitialisation	uniform (`uni`)
	–	–	–	midpoint to target	midpoint target	midpoint target	halfway (`HVB`)
	–	–	midpoint to base	midpoint to base	–	midpoint base	–
	–	exponentially confined, spread	–	–	–	–	COTN
	–	–	–	–	other SDIS[4]	other SDIS[5]	–
Vector-wise SDIS	–	shrink[6]	scaled mutant[7]	projection to midpoint or base[8]	centroid $K+1$ [9]	–	–
	–	inverse parabolic (confined, spread)	–	–	–	–	–

[2] Unimodal test functions (Ellipsoidal, Schwefel, Ackley, Rosenbrock)

[3] For $n = 20$ and $n \in \{50, 100, 200, 300, 500\}$, respectively

[4] Conservatism, resampling, evolutionary

[5] Death-penalty, resampling, boundary-transformation, rand-base, projection-to-midpoint, projection-to-base

[6] Linear combination between the trial and base elements

[7] Linear combination between the trial and a reference element in the feasible region

[8] Linear combination between the trial and the domain midpoint or base element

mechanisms for dealing with infeasible solutions, which might appear in the literature under different names (see e.g. Table 1 and Section 2.5).

Unlike 'vector-wise' SDIS, where vector operations act on the totality of the components to obtain a feasible point, they operate 'per-component' by modifying only the infeasible coordinates while keeping the feasible ones unchanged. To distinguish between the two working mechanisms, we refer to the latter as 'coordinate-wise' SDIS.

It is also worth clarifying that, in the context of DE, `HVB` acts on the `trial` individual by using the `target` individual as its feasible counterpart. Hence, before calling the objective function, any infeasible design variable of the `trial` solution would get replaced with a feasible value located halfway from the position of the corresponding component in the `target` solution and the violated bound (upper or lower). In this light, this SDIS can be seen as a coordinate-wise counterpart of the 'projection to midpoint' repair strategy for DE used in (Biedrzycki et al., 2019), where the `mutant` vectors are manipulated into feasible `trial` solutions as further discussed in Section 2.5.

2.5 State-of-the-art on strategies of dealing with infeasible solutions

Carried out as a part of the current paper, a review on recent publications which propose new or improved DE variants revealed that only a small proportion of papers consider the strategy of dealing with infeasible solutions as a mechanism influencing the search process and describe explicitly the used SDIS. As is illustrated in Table 2, several categories of papers have been identified in case of DE.

On one hand, most of the papers do not provide access to the source code containing the implementation of the proposed algorithm, which, on its own, implies reproducibility issues. In this case, the only source of information is the algorithm description provided in the paper. In the few cases when the SDIS is explicitly specified in the paper, no strong motivation for its choice is provided (simplicity or popularity, being typically mentioned) and its influence on the algorithm behaviour is not discussed. Another category contains papers that present variants of a state-of-the-art method (e.g. `JADE` or `SHADE`) and the reader might infer, in the absence of an explicit statement on SDIS, that the strategy used in the original algorithm (e.g. the so-called

Table 2: Summary of ways in which infeasible solutions aspect is addressed in recent works proposing new DE variants

Source code not available	**1.** SDIS is mentioned in the paper	(Deng, 2020), (Cheng et al., 2021), (Mohamed et al., 2021): `uniform`; (Brest et al., 2019): `mirror`; (Liu et al., 2019), (Stanovov et al., 2020): `HVB`; (Zhan et al., 2020), (Zhao et al., 2020): `saturation`; (Deng et al., 2022): `midpoint-base`.
	2. The proposed algorithm is derived from `SHADE`, `JADE` and it might be assumed that SDIS is inherited	(Awad et al., 2018), (Viktorin et al., 2019), (Cheng et al., 2020), (Meng et al., 2020), (Yi et al., 2021), (Zhong and Cheng, 2021), (Kumar et al., 2022) (Zuo and Guo, 2022): `HVB`.
	3. The proposed algorithm is a new or an enhanced DE variant and SDIS is not mentioned	(Tian and Gao, 2019), (Choi et al., 2020), (Mousavirad and Rahnamayan, 2020), (Sun et al., 2020), (Wang et al., 2020), (Zhou et al., 2020), (Mousavirad et al., 2021), (Song and Li, 2021).
Source code available	**1.** SDIS is mentioned in the paper	(Mohamed, 2018), (Mohamed and Mohamed, 2019): `uniform`;
	2. SDIS is not mentioned in the paper, but used in implementation	(Tomczak et al., 2020): `saturation`.

`midpoint-to-target`, or `HVB` in the terminology of this paper) is used in the proposed variant. However, this is just the guess of the reader and the reproducibility of the results is at least questionable. The third category of papers includes descriptions of algorithms without any specification on how the solutions that violate the bound constraints have been treated.

On the other hand, there are papers for which the source code is made available, thus even if the SDIS is not described in the paper it can be identified in the code. However, the reasoning behind choosing one strategy over the other ones is still missing. Similar remarks are presented in a recent paper (Kadavy et al., 2022) where the importance of considering the bound constraint handling method as a specific component of metaheuristic optimisation algorithms is emphasised. The analysis was conducted on algorithms participating in CEC2017 and CEC2020 competitions on single-objective continuous optimisation and revealed that for only one-third of algorithms (8 out of 24) the method of handling the bound constraints was explicitly specified. The experiments also showed that the performance of some of the competing algorithms could be improved by changing the bound constraint handling method, leading even to some changes in the final ranking of the competition.

It should be mentioned that there are several works devoted to the comparative analysis of different strategies to deal with infeasible solutions in the context of various metaheuristics: CMA-ES (Wessing, 2013; Biedrzycki, 2019), Particle Swarm Optimisation (Cheng et al., 2011; Helwig et al., 2013; Juárez-Castillo et al., 2017; Oldewage et al., 2018), Differential Evolution (for DE see a summary in Table 1).

In relation to Differential Evolution, the first paper presenting a comparison between the performance of various SDISs applied to `DE/rand/1/bin` (Arabas et al., 2010) analyse the following strategies: `sat` (referred to as `projection`), `tor` (referred to as `wrapping`), `uni` (referred to as `reinitialisation`) and `mir` (referred to as `reflection`). The main observation is that the choice of SDIS might have an influence on the DE performance but the amount of impact depends on the problem characteristics (e.g. position of the optimum and problem size): (i) `sat` and `mir` work well when the optimum is near the bounds; (ii) for small size problems (e.g. $n = 10$) the number of corrected elements is not significantly influenced by the used SDIS and there are no significant differences between the performance of various strategies; (iii) for larger size problems (e.g. $n = 30$) a higher effectiveness has been observed for `sat` and `mir` when compared with `uni`.

In the study (Padhye et al., 2015) addressing the influence of SDIS on Particle Swarm Optimisation, Differential Evolution (`DE/best/1/bin`), and Genetic Algorithms, it is stated that deterministic methods, for instance `sat`, lead to a loss in population diversity, while `uni` (referred in the original paper as `random-reinitialisation`) loses useful information carried by the current population. The main remark on the performance of DE combined with a SDIS is that when the optimum is near the midpoint of the feasible domain there is no significant difference between the impact of various strategies. On the other hand, when the optimum is close to the boundary then a parameterised vector-wise non-deterministic strategy (`inverse-parabolic`) behaves the best with the experimental setup of the paper.

Currently, the most extensive study on bound constraints handling for standard DE is (Biedrzycki et al., 2019), which presents experimental results based on the CEC 2017 benchmark suite. The authors analyse the impact of different strategies of dealing with infeasible solutions (penalty functions, repairing methods, feasibility preserving mutation) on the dynamics of the population, convergence speed, and global optimisation efficiency. The main insights reported in (Biedrzycki et al., 2019) are: (i) the greatest influence on the mean and variance of the mutant population distribution is induced by `uni` and `tor` SDIS; (ii) the performance sensitivity of standard DE to SDIS is higher for larger size problems; (iii) the adaptive variants (e.g. `JADE`, `SADE` and `jSO`) are less sensitive to the choice of SDIS than non-adaptive DE; (iv) the influence of a SDIS depends on the DE variant with which it is combined, but overall the best behaviour is induced by `midpoint` strategies (which use values between the corresponding component of the target or base vector and the violated bound) and by `mir`. The authors of (Biedrzycki et al., 2019) consider that the results of the experimental analysis can be explained by the fact that a SDIS with better performance leads to a lower discrepancy between the distributions of the before and after correction populations. It should be emphasised that best performance is obtained when the mutant construction is repeated by sampling new parents until a feasible trial vector is obtained. A similar conclusion has been reported also in (Kreischer et al., 2017) where `DE/target-to-best/1/bin` algorithm (with $F = 0.8$ and $C_r = 0.9$) is combined with the same SDISs mentioned above and tested on CEC 2017 benchmark. The authors of (Kreischer et al., 2017) also recommend `mir` and `projection` to an interior point of the feasible region as strategies which perform well in the case of CEC 2017 benchmark, followed by `HVB`.

An experimental study on the influence of nine strategies is presented in (de-la-Cruz-Martínez and Mezura-Montes, 2020) aiming to deal with bound constraints when combined with the so-called Deb feasibility rules (Deb, 2000) to solve four real-world constrained optimisation problems related to mechanical design. The analysed strategies are `midpoint-target` (`HVB`), `reflection` (`mir`), `projection` (`sat`), random-scheme (uni), full reinitialisation (all components, including the feasible ones, are randomly reinitialised), conservatism (the trial vector is just discarded), resampling (a new mutant is constructed by using other randomly selected parents), evolutionary [10] (the infeasible component is replaced with a convex combination between the violated bound and the corresponding component from the best element in the population), centroid K+1 (average of a set including an element selected from the population and K other elements obtained by applying uni to the infeasible components). The main conclusion is that the influence of the SDIS is highly dependent on the problem to be solved but overall, sat proved to be the most effective strategy.

Similar to the setup in (Biedrzycki et al., 2019), (Boks et al., 2021) have investigated the effect of SDIS on performance on the single-objective noiseless version of the BBOB benchmark (Finck et al., 2010) in DE for a wide selection of operators, in a fully modular fashion. The considered mutation operators are rand/1, best/1, target-to-best/1, best/2, rand/2, target-to-best/2, target-to-*p*best/1, rand/2/dir, NSDE, trigonometric, 2-opt/1, 2-opt/2, proximity-based rand/1, and ranking-based-target-to-*p*best/1. These are combined with bin and exp crossovers with SHADE-based adaptation of control parameters. The resulting DE variants are tested with a wide range of SDIS operators, namely death-penalty, resampling, reinitialisation, projection, reflection, wrapping, boundary-transformation, rand--base, midpoint-base, midpoint-target, projection-to-midpoint, projection-to-base, and conservatism. This paper appears to be the only analysis of performance dependency on SDIS on the BBOB benchmark. The main conclusions of (Boks et al., 2021) are: (i) no SDIS appears to be optimal for all DE configurations considered; (ii) SDIS ranks differ greatly between configurations and BBOB function groups; (iii) to some extent, the best SDIS tends to depend on crossover. As a rule of thumb, for similar setups, practitioners are therefore advised to consider conservatism for exp crossover and reinitialisation for bin crossover as they perform best with

[10]Name as given by (de-la-Cruz-Martínez and Mezura-Montes, 2020)

many configurations; such policy, however, does not always give the optimal result. Optionally, `midpoint-target` in `bin` configurations rarely performs best but always performs well and `projection-to-midpoint` is a reliable second option for `exp` configurations. Finally, over all cases considered, `resampling` SDIS has been deemed successful the highest number of times.

For DE implementations incorporated in various popular open-source *libraries*, the most common SDIS is `uni` (used in SciPy [11], PyMOO [12] and PAGMO [13]), followed by `sat` (in the MOEA framework [14], PyADE [15]) and `mir`, in PyMOO. Finally, a *notable exception* is a highly modular AutoDE library [16] based on the aforementioned paper (Boks et al., 2021) which provides many standard DE configurations and a hyper adaptive version of SHADE (Boks, 2021), all with 13 SDIS variants (see the list above).

3 Search direction

Any heuristic iterative optimiser can be considered an *adaptive sampler* which is guided according to some logic by differences in values of the objective function (or derivatives thereof) evaluated in the previously sampled points. It, therefore, makes sense to consider a 'path' taken by an optimiser in the search domain or, more practically, a sequence of search directions over sampled points. This section defines the search direction induced by the DE operators and proposes to quantify the influence of a SDIS by computing the cosine similarity between the unconstrained search direction and the search direction resulting after applying a SDIS.

3.1 Definition of search direction in Differential Evolution

While it is easy to define the search direction for iterative single-solution methods, such a task gets excessively complicated in the case of general population-based iterative heuristic optimisers where multiple solutions steer the generation of subsequent solutions. However, Differential Evolution (see Section 2.2) lends itself to such analysis

[11]Package `scipy.optimize.differential_evolution` (Virtanen et al., 2020)
[12]`https://pymoo.org/algorithms/soo/de.html`
[13]`https://esa.github.io/pagmo2/docs/cpp/algorithms/de`
[14]`http://moeaframework.org/`
[15]`https://github.com/xKuZz/pyade`
[16]`https://github.com/rickboks/auto-DE` (Boks et al., 2021; Boks, 2021)

easier thanks to its survivor selection mechanism based on a '1-to-1 spawning' logic. This means that every new solution updates its direction from the direction of a parent and the whole population represents a repeatedly updated *ensemble* of search directions with a clear 'inheritance' scheme. Note that this happens despite the fact that new solutions also incorporate information from others in the population apart from their direct parent, and implicitly capitalise on the information contained in the population, following the spirit of population-based heuristic optimisation (Prügel-Bennett, 2010).

In this context, we consider for each individual in the population, interpreted as a target individual, a *search direction* which is defined as the difference between the corresponding `trial` and `target` individuals. When an unfeasible trial individual is corrected by applying a SDIS, the search direction will be altered. Changes in such 'sequence of search directions' can then be measured via, e.g. cosine similarity.

3.2 Cosine as a measure of similarity between search directions

The Cosine Similarity (CS) between non-zero vectors $\mathbf{v_1}$ and $\mathbf{v_2}$ is defined as their 'normalised' inner product:

$$\mathrm{CS}(\mathbf{v_1}, \mathbf{v_2}) = \frac{\mathbf{v_1}^{\mathrm{T}}\mathbf{v_2}}{\|\mathbf{v_1}\|\|\mathbf{v_2}\|} \tag{2}$$

thus not depending on their magnitudes and registering only differences based on the angle θ between them. For this reason, it can be seen as an angular distance, and can be used to determine whether two directed vectors $\mathbf{v_1}$ and $\mathbf{v_2}$ are pointing to the same oriented direction. Indeed, let us note that $\mathrm{CS}(\mathbf{v_1}, \mathbf{v_2}) = \cos(\theta) \in [-1, 1]$ and observe that:

- $\mathrm{CS}(\mathbf{v_1}, \mathbf{v_2}) = 0 \iff$ $\mathbf{v_1}$ and $\mathbf{v_2}$ are orthogonal (i.e. the most dissimilar);
- $\mathrm{CS}(\mathbf{v_1}, \mathbf{v_2}) = 1 \iff$ $\mathbf{v_1}$ and $\mathbf{v_2}$ are parallel (i.e. aligned and pointing to the same oriented direction);
- $\mathrm{CS}(\mathbf{v_1}, \mathbf{v_2}) = -1 \iff$ $\mathbf{v_1}$ and $\mathbf{v_2}$ are anti-parallel (i.e. have opposite oriented directions).

In this study, we detect changes in the *search direction* during the evolution in DE algorithms through the CS between two vectors, namely:

- $\mathbf{d}$: obtained as the difference between the trial individual (before a SDIS is applied) and the target vector - i.e. the `target-to-trial` directed vector,
- $\mathbf{d_c}$: obtained as the difference between the corrected trial individual (after a SDIS is applied) and the target vector - i.e. the `target-to-feasibleTrial` directed vector,

to *observe if the employed SDIS is responsible for a change in the search direction*. In a single run, CS values are computed after a new trial individual is available, thus generating a sequence of angular distances over time where the length of such sequence is equal to the computational budget used minus the population size (to account for the initial population). Sequences obtained across multiple runs for the same algorithmic configuration are then aggregated for analysis purposes as indicated in Section 5.2. For feasible solutions, CS values are *not computed* and thus excluded from the analysis.

3.3 Strategy of dealing with infeasible solutions as source of search disruptiveness

In the case of an unconstrained search, Differential Evolution can generate, for given values of the control parameters, only a finite set of trial individuals which define the corresponding set of search directions. When a SDIS is incorporated in DE, these search directions can be altered through the introduction of new moves/perturbations, hence a SDIS can be viewed as a source of disruptiveness in the DE search process. When analyzing the impact of a SDIS on the search process there are at least *two questions* that arise: (i) how much is the search process altered? (ii) does a SDIS have a beneficial or a detrimental influence on the performance of the search?

The amount of disruptiveness depends both on the number of components in the trial individuals which are corrected by the SDIS and on the characteristics of the SDIS. One way to quantify the influence of the SDIS on the search process is to compute the cosine between the uncorrected and the corrected search directions, as it is easy to compute and can provide useful information even for high dimensions. When this value is closer to one, more of the search direction is preserved. Besides the influence on the search direction, different SDISs lead to different positions of the corrected individuals and, as the chance of generating such individuals through the DE mutation (based on the current population distribution and values of the control parameters) is smaller, the disruptive character of the SDIS can be considered more significant.

The second question is more difficult to answer, because the interference between the SDIS and the DE mechanisms can hinder but also help the search. At a first sight, at least the coordinate-wise SDISs are characterised by inappropriate usage of the information contained in the population. Indeed, these alter the self-referential search direction induced by the difference-based mutation and decouples the components (with impact on the ability of some DE variants to be rotationally invariant (Bujok et al., 2014; Caraffini and Neri, 2019)). From this point of view, a vector-wise SDIS, as that proposed in (Kreischer et al., 2017), can be considered less disruptive.

On the other hand, interfering with the DE search might be beneficial at least with respect to: (i) generation of trial individuals which would otherwise not be generated by DE operators, thus increasing the pool of candidate solutions (this might be particularly useful in the case of small populations); (ii) influence on the population diversity by including components or individuals which do not fully rely on differences (this might be useful to avoid premature convergence or stagnation which are two of the main causes for lack of performance of DE). So we can say that a *SDIS might turn into an additional diversity increasing mechanism.* Some of these aspects are discussed in Section 4.4 and illustrated further in Section 5.3, but there are some more complex issues related to the control of diversity which are not discussed here.

4 Theoretical analysis of strategies of dealing with infeasible solutions

Under the usual assumption that the scale factor, F, is less than one, all mutant vectors generated by `DE/rand/1` will have the components inside the extended domain $[2a_i - b_i, 2b_i - a_i]$, thus the infeasible components will belong to $[2a_i - b_i, a_i) \cup (b_i, 2b_i - a_i]$, where $f : \mathbf{D} = \times_{i=1}^{n} [a_i, b_i] \to \mathbb{R}$.

To quantify the influence of a SDIS on the search dynamics, several quantities can be taken into account: (i) the number of corrected components (in the case of coordinate-wise strategies); (ii) the difference between the infeasible trial element and the corrected one; (iii) the (dis)similarity between the search direction as it would be in an unconstrained search and the corrected search direction; (iv) the impact of SDIS on the population diversity. Different SDISs can behave differently with respect to these aspects. A theoretical analysis, even if conducted under some simplifying assumptions, might lead to some insights which could explain empirical observations or provide guidelines for selecting a SDIS.

The results presented in this section correspond to `DE/rand/1/`$\star$ [17] and are obtained under some necessary *simplifying assumptions*: (i) absence of selection pressure (this is equivalent with using a *flat fitness function*, i.e. all new trial individuals are accepted as soon as they are generated); (ii) the analysis of population diversity is conducted coordinate-wise; (iii) the distribution of the individuals in the current population is close to the uniform one. The last assumption is used only for the estimation of the probability to generate infeasible components and for the estimation of the variance of the population of individuals which are corrected by using `mir`.

Furthermore, results discussed in this section under the aforementioned simplifying assumptions are *contrasted* with empirical results on benchmark functions in Sections 5, 6 which are generally free of such assumptions. Formal proofs for statements in this section are provided in Appendix A.

4.1 Estimation of bound violation probability

The number of infeasible components in a `DE/rand/1` mutant depends on the mutation probability (p_m) and on the probability, called bound violation probability (p_v), to generate by mutation an outside the bounds component. The mutation probability depends on C_r, on the problem size, n, and on the crossover type (Zaharie, 2009). On the other hand, the bound violation probability depends on the mutation type and on the value of the scale factor F. In the particular case when the distribution of the current population is close to the uniform distribution, the probability that a component

[17] Symbol $\star$ means that either `bin` or `exp` crossover can be used.

violates the bounds is close to $F/3$ (Zaharie and Micota, 2017). The distribution of the trial population is influenced both by the mutation operator and by the used SDIS. One question is whether the use of a SDIS in the current generation increases or decreases the risk of generating infeasible components in the next generation. This question is rather difficult to answer in the general case, but it can be addressed in the case of `sat` strategy. In this case, the current population, P, consists of three subpopulations: P_w (elements within the bounds), P_{lb} (elements on the lower bound), and P_{ub} (elements on the upper bound). If $p_v(g)$ denotes the bound violation probability corresponding to generation g (for the first generation population it is considered that $p_v(1) = F/3$) then it can be proved (Proposition 1 in Appendix A.1), under the assumption that P_w remains almost uniformly distributed, that

$$p_v(g+1) \simeq p_v(g)/2 + (1-p_v(g))(p_v^2(g)F/2 + (1-p_v^2(g))F/3) \tag{3}$$

This sequence of probabilities, $p_v(g)$, converges to a value between $F/3$ and $2F/3$ (see Figure 12 in Appendix A.1). This suggests that, in the absence of selection pressure, `sat` increases the risk of generating infeasible components, but not by a large amount. This is confirmed by the empirical results presented in Section 5.4, Figure 4.

4.2 Difference between infeasible and corrected elements

The simplest way to quantify the impact of a SDIS is simply to compute the Euclidean distance between an infeasible trial individual, z, and its corrected version, z^{corr}. Using Eq. (4), where $c(y_i)$ denotes the i-th component of the mutant transformed by applying a SDIS, it follows that the expected value of $\|z - z^{\text{corr}}\|^2$ is $p_m p_v \sum_{i=1}^{n}(y_i - c(y_i))^2$.

$$z_i = \begin{cases} x_i & \text{with probab. } 1-p_m \\ y_i & \text{with probab. } p_m \end{cases} \qquad z_i^{\text{corr}} = \begin{cases} x_i & \text{with probab. } 1-p_m \\ y_i & \text{with probab. } p_m(1-p_v) \\ c(y_i) & \text{with probab. } p_m p_v \end{cases} \tag{4}$$

For a given component, the difference $(y_i - c(y_i))^2$ is obviously the smallest in the case of `sat`. If $0 < y_i - b_i \leqslant (b_i - a_i)/2$ or $0 < a_i - y_i \leqslant (b_i - a_i)/2$ then the difference is smaller for `mir` than for `tor`. This always happens if $F \leqslant 0.5$. In the case of `uni`, as the corrected value can be anywhere in $[a_i, b_i]$, the difference can take any value in $(0, 2(b_i - a_i))$. However, the bound violation probability induced by `sat` is usually higher than in the case of the other SDISs. This means that the expected value of the Euclidean distance corresponding to `sat` is not necessarily the smallest one.

4.3 (Dis)similarity between search directions

The similarity between the DE search direction, i.e. $d = z - x$ (difference between the infeasible trial individual, z, and the target individual, x) and the corrected search direction, $d_c = z^{\text{corr}} - x$, can be analysed using the cosine of the angle between d and d_c. In the case of `sat`, the components of the corrected search direction, $d_{\texttt{sat}}$, have the same sign as the components of the DE search direction (either $z_i < a_i = c_{\texttt{sat}}(z_i) \leqslant x_i$ or $z_i > b_i = c_{\texttt{sat}}(z_i) \geqslant x_i$, where $c_{\texttt{sat}}$ denotes the transformation corresponding to `sat`). Thus, for `sat` the cosine similarity is always greater than 0. The highest value of the similarity is obtained when z^{corr} is a convex combination between x and z, which means that the SDIS does not alter the search direction.

The symmetry between the corrected solutions obtained by `mir` ($c_{\texttt{mir}}$) and `tor` ($c_{\texttt{tor}}$), i.e. $c_{\texttt{mir}}(z_i) + c_{\texttt{tor}}(z_i) = a_i + b_i$ allows us to prove (see Proposition 2 in Appendix A) that when $F \leqslant 0.5$ (which ensures that $d^T d_{\texttt{mir}} \geqslant d^T d_{\texttt{tor}}$) and x and $c_{\texttt{mir}}(z)$ are in the same quadrant, i.e. $(c_{\texttt{mir}}(z_i) - (a_i + b_i)/2)(x_i - (a_i + b_i)/2) \geqslant 0$ for $i = \overline{1, n}$ (which ensures that $\|d_{\texttt{mir}}\| \leqslant \|d_{\texttt{tor}}\|$) then $\cos(d, d_{\texttt{mir}}) \geqslant \cos(d, d_{\texttt{tor}})$. When x and $c_{\texttt{mir}}(z)$ are not in the same quadrant, then it is not necessary that $\|d_{\texttt{mir}}\| \leqslant \|d_{\texttt{tor}}\|$, thus the relationship between $\cos(d, d_{\texttt{mir}})$ and $\cos(d, d_{\texttt{tor}})$ cannot be inferred so easily.

For the other SDISs, it is difficult to prove statements on search directions in the general case, since the target value, x, can be placed anywhere in the feasible domain. However, in the case where only one component is corrected and the Euclidean norm of the uncorrected search direction is not fully determined by the infeasible component, `sat` preserves the search direction better than any other SDIS that generates values inside the open domain, i.e. (a_i, b_i). More specifically, if k denotes the index of the infeasible component (e.g. $z_k > b_k$) and if $\|d\|^2 = \sum_{i=1}^{n} (z_i - x_i)^2 \geqslant 2(z_k - x_k)(z_k - b_k)$

then $\cos(d, d_{\texttt{sat}}) \geqslant \cos(d, d_{other})$ (see the proof of Proposition 3 in Appendix A). The above constraint on $\|d\|^2$ might be violated in the case where C_r is small, leading to very few mutated components (in the extreme case only one component is mutated, and this is also infeasible) and F is large, leading to a large deviation of z_k with respect to x_k.

4.4 Influence of strategy of dealing with infeasible solutions on population diversity

The population diversity can be quantified using the average variance of the population computed coordinate-wise, i.e. $\text{var}(X) = \frac{1}{n}\sum_{i=1}^{n}\text{var}(X_i)$ and $\text{var}(X_i) = \dfrac{1}{N}\sum_{j=1}^{N}(x_i^j - \overline{X}_i)^2$ where $\overline{X}_i$ denotes the average of the i^{th} component values over the entire population of N individuals. In the following, the analysis is performed for one component; thus, index i is skipped. In the case of `DE/rand/1` mutation, the expected value of the corrected trial population variance (after applying SDIS, $c(Z)$) depends on the variance of the current population, X, as given (Zaharie and Micota, 2017), in Eq.5:

$$\mathbb{E}[\text{var}(c(Z))] = \alpha(p_m, p_v, N, F)\cdot\text{var}(X) + \beta(p_m, p_v, N, SDIS). \tag{5}$$

The first coefficient, $\alpha(p_m, p_v, N, F)$ is not influenced by the used SDIS, depending only on the mutation probability (p_m), bound violation probability (p_v), population size (N) and the scale factor (F). The influence of SDIS is incorporated into the free term, which depends on the average and variance of the corrected values, as is specified in Eq. 6, where $\overline{X}$ denotes the midpoint of the current population, mean($SDIS$) and var($SDIS$) denote the average and variance of the individuals corrected using a specific SDIS, respectively.

$$\begin{aligned}\beta(p_m, p_v, N, SDIS) \quad &= \quad p_m p_v(1 - p_m p_v)\frac{N-1}{N}(\overline{X} - \text{mean}(SDIS))^2 \\ &\quad +p_m p_v\left(1 - \frac{1 - p_m p_v}{N}\right)\text{var}(SDIS)\end{aligned} \tag{6}$$

In the case of `uni` strategy, mean(`uni`) $= (a+b)/2$ and var(`uni`) $= (b-a)^2/12$. In the case of `sat` strategy one can consider, in the absence of the selection pressure, that the set of corrected individuals follows a Bernoulli distribution with values a and b and corresponding probabilities $p_{lb} + p_{ub} = 1$. Since, in the case of a flat function, there is no incentive to bias the population towards one of the bounds, one can consider $p_{lb} = p_{ub} = 1/2$. In this case, we obtain mean(`sat`) $= (a+b)/2$ and var(`sat`) $= (b-a)^2/4$. This suggests that, at least under these simplifying assumptions, the influence of SDIS on population variance is greater in the case of `sat` than in the case of `uni` strategy, since the second term of $\beta(p_m, p_v, N, SDIS)$ is greater in the case of `sat`.

For `mir` and `tor` strategies, the distribution of the population of corrected individuals is directly related to the distribution of the uncorrected trial population (Z). In the case of `mir`, a corrected individual, $c_{\texttt{mir}}(z)$, is $2b - z$ with probability p_{ub} and $2a - z$ with probability p_{lb}. Since for `DE/rand/1` the expected mean of the mutant population ($\mathbb{E}[\overline{Z}]$) is the same as the mean of the current population ($\overline{X}$), it follows that mean(`mir`) $= (a+b) - \overline{X}$. Thus, $(\overline{X} - \text{mean}(\texttt{mir}))^2$ is close to $4\left(\overline{X} - \frac{a+b}{2}\right)^2$, suggesting that the first rhs term in Eqs.6 is four times larger in the case of `mir` strategy than in the case of the `sat` and `uni` strategies. However, if $\overline{X} - \frac{a+b}{2}$ is small, the influence of this term is also small. On the other hand, the variance of the population of individuals that violate one of the bounds and are corrected using the `mir` strategy is expected to be close to $F^2/10 - F/4 + 1/4$ (under the assumptions that $a = 0$ and $b = 1$ and as long as the distribution of the current population, X, is close enough to the uniform distribution), which is less than $1/4$ (variance of the individuals corrected by `sat`) but greater than $1/12$ (variance of the individuals corrected by `uni`). For details, see the proof of Proposition 4 in Appendix A.

Thus, the theoretical analysis suggests that the greatest impact on population diversity is induced by `sat` followed by `mir` and then by `uni`. Due to the symmetry between `mir` and `tor` (as $c_{\texttt{mir}}(z) + c_{\texttt{tor}}(z) = a + b$), the population of corrected individuals has the same variance; thus, it is expected that these two strategies have the same influence on population diversity.

When `COTN` strategy is used, the corrected mutant is calculated as $c_{\texttt{COTN}}(z) = a +$

$|\xi|\sigma$, in the case of a lower bound violation and as $c_{\texttt{COTN}}(z) = b - |\xi|\sigma$, in the case of a violation of the upper bound, where $\xi \sim N(0,1)$ and $\sigma = (b-a)/3$. The probability distribution of the random variable corresponding to the corrected individuals is given in Eq. 7, where $R = a$ if the lower bound is violated, and $R = b$, if the upper bound is violated:

$$f_{\texttt{COTN}}(v) = \frac{2}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(v-R)^2}{2\sigma^2}\right). \tag{7}$$

It should be mentioned that when using $c_{\texttt{COTN}}$ as defined above, there is a nonzero probability to generate individuals which violate the opposite bound, as the support of f_{COTN} is $[a,\infty)$ when $R = a$ and is $(-\infty, b]$ when $R = b$. If σ is not large then this probability is small; e.g., if $\sigma = (b-a)/3$, the probability of violating the opposite bound is 0.00269. In the experimental analysis, this SDIS is iterated until a feasible value is obtained. Under the assumption that the probabilities to violate the bounds are equal ($p_{lb} = p_{ub} = 0.5$), the variance of the mixture of corrected individuals (corresponding to $R = a$ and $R = b$ in Eq. 7, respectively) is $0.095(b-a)^2$ (for details, see Proposition 5 in the Appendix) which is greater than $(b-a)^2/12 = 0.083(b-a)^2$. This suggests that `COTN` has a slightly greater impact on population variance than `uni`.

5 Experimentation on f_0

Understanding the 'behaviour' of a heuristic algorithm during the search for near-optimal positions is challenging. As the sampled fitness landscape of the problem at hand is the main driving force of this process, decoupling its effect on the internal dynamics of the candidate solutions is key to observing how the algorithm's working logic operates in figuring out promising search directions to follow. The f_0 function of Eq. 8 was first proposed in (Kononova et al., 2015) to serve such a purpose.

$$\mathbf{f_0} : [\mathbf{0},\mathbf{1}]^{\mathbf{n}} \to [\mathbf{0},\mathbf{1}], \text{ where } \forall \mathbf{x}, \mathbf{f_0}(\mathbf{x}) \sim \mathcal{U}(\mathbf{0},\mathbf{1}). \tag{8}$$

In other words, values of the objective function f_0 are sampled from a uniform distribution which is seeded before the start of each optimisation run. It should be noted that a similar function has been independently proposed (Cleghorn and Engelbrecht,

2014) for investigating the convergence of the particle swarm optimisation (PSO) algorithm – as an example of a degenerate function which simultaneously warrants against stagnation of PSO and provides a difficult convergence scenario.

Because of its stochastic nature, f_0 separates otherwise highly interconnected effects from the fitness landscape and the location of the optima, thus being suitable for analysing structural implications to the search. A previous study used f_0 to understand how often solutions are generated outside the search space in classic DE variants (Kononova et al., 2021). Relevant observations are reported in Section 5.4. Other studies used it to find structural biases in DE variants (Caraffini and Kononova, 2019; Caraffini et al., 2019; van Stein et al., 2021; Vermetten et al., 2022d), as well as several other algorithmic frameworks for heuristic optimisation (Kononova et al., 2020b,a; Vermetten et al., 2021). A summary is reported in Section 5.5. In this work, we relate to these aspects and further employ f_0 to collect the CS angular distances in the two most known DE configurations and study their distribution.

Where possible, the results on the function f_0 discussed in this section are put in contrast to the theoretical conclusions obtained under a number of simplifying assumptions made in Section 4.

5.1 Setup for the experimentation on f_0

The two well-known `DE/rand/1/bin` and `DE/rand/1/exp` algorithms are executed on f_0 function in $n = 30$ dimensions (dimensionality consistent with previous studies on f_0, thus allowing for a direct comparison) for a fixed number of $10000 \cdot n$ function evaluations. This is done for multiple configurations of the two DE variants equipped with SDIS from the set $\{\texttt{COTN}, \texttt{mir}, \texttt{sat}, \texttt{tor}, \texttt{uni}, \texttt{HVB}\}$ (see Section 2.4), a specific population size $N \in \{5, 20, 100\}$ and a pair of control parameters F and C_r.

To practically identify these configurations in our analysis we use the notation $\texttt{DE/rand/1/}\star\texttt{-p}N$. As each configuration is tested over a wide range of $F - C_r$ pairs, these are not included in the notation, but are reported more effectively in the graphical results. Note that the same set of scale factor values, i.e. $F \in \{0.05, 0.285, 0.52, 0.755, 0.99\}$, is considered for the `bin` and `exp` DE variants, which offers a good discretisation of the space of its typical admissible values. On the contrary, two different sets of values are used for C_r depending on the crossover operator used, as indicated below:

- $C_r^{bin} \in \{0.05, 0.285, 0.52, 0.755, 0.99\} \cup \{0.0891, 0.1283, 0.1675, 0.2067, 0.2458\}$,
- $C_r^{exp} \in \{0.05, 0.285, 0.52, 0.755, 0.99\}$,

where the admissible range $(0, 1]$ is first discretised equally for the two crossover strategies and a further set is added to better cover the range $(0.05, 0.285)$ when the operator `bin` is used. The latter additional range is of interest to our analysis, as it allows us to detect the nature of changes (smooth vs sudden) in the algorithm behaviour, measured here in terms of cosine similarity between search directions (as indicated in Section 3.2), when only one or very few components are inherited and corrected. It should be noted that doing this while employing the `exp` crossover operator would be irrelevant. Indeed, with such low C_r values, this operator would be quite unlikely to exchange any component from the `mutant` to the `target` on top of the one that is necessarily replaced by design; for clarifications on the behaviour of the `exp` operator, see (Caraffini et al., 2019; Kononova et al., 2021; Vermetten et al., 2022d).

In total, the `DE/rand/1/exp` configurations obtained from combining the 6 SDIS operators, the 3 population sizes, the 5 scale factor values, and the 5 crossover rate values (that is, the $6 \cdot 3 \cdot 5 \cdot 5 = 450$ configurations), plus those obtained with similar settings but with 5 more C_r values for `DE/rand/1/bin` (that is, $6 \cdot 3 \cdot 10 \cdot 5 = 900$), results in 1350 optimisation processes. Each DE configuration is executed 30 times to produce robust results over 30 independent runs.

Relevant information stored during each run includes CS measure values for solutions where SDIS has been applied, percentages of infeasible solutions (see Section 6.4), population and fitness diversity measures, etc. This experimentation has been carried out with the SOS software platform (Caraffini and Iacca, 2020), whose source code is available on its GitHub repository, where a permanent release of the current state of the SOS platform is also available [18]. This is accompanied by detailed clarifications on

[18]`https://github.com/facaraff/SOS/releases/tag/ECJ-ReproducibilityInEC`

how to find software classes within the platform and how to reproduce the entire data set used, which we have stored in the Zenodo repository (Vermetten et al., 2022a) along with the full sets of static versions of all processing scripts.

5.2 Analysis of results on distributions of cosine similarity

To analyse the distribution of cosine similarity for infeasible solutions when using different SDIS, we can use the Empirical Cumulative Distribution Function (ECDF), which shows for each value of x what fraction of infeasibility corrections had a CS of at most x.

In Figure 2, we show the ECDF of CS values, for each SDIS, based on the values F and C_r, for *all infeasible* solutions generated during full runs of `DE/rand/1/bin-p100` on f_0. In this figure, SDIS with a larger disruptiveness will have a curve that is closer to the upper left corner of the plot, so larger areas under the ECDF correspond to smaller values of the cosine between unconstrained and corrected search directions, i.e. larger amounts of disruptiveness. When comparing two strategies, in the case when $ECDF(SDIS_1) \geqslant ECDF(SDIS_2)$ then it can be proved (see Proposition 6 in Appendix A.5) that it is more likely that the cosine values corresponding to $SDIS_1$ are smaller than those corresponding to $SDIS_2$.

We see in Figure 2 that even though the shape of the curves can change significantly based on the parameter settings used, the ordering remains consistent: `tor` is the most disruptive, followed by `uni`, `COTN` and `mir`, while `HVB` and `sat` are the least disruptive. This ordering is also preserved when changing the population size or the crossover type for `DE/rand/1` – those figures are included in the Figshare repository of this article (Vermetten et al., 2022b).

Relation to theoretical results. The experimental results illustrated in Figure 2 lead to remarks that are *in line* with the results derived in Section 4.3: (i) `sat` is more likely to lead to the highest value of cosine similarity (provable in the case where only one component is corrected - this could be related to small values of F and C_r); (ii) in the case when $F \leqslant 0.5$ it is more likely that `tor` induces a higher disruption than `mir` (provable when the individual obtained by mirroring and the target individual are in

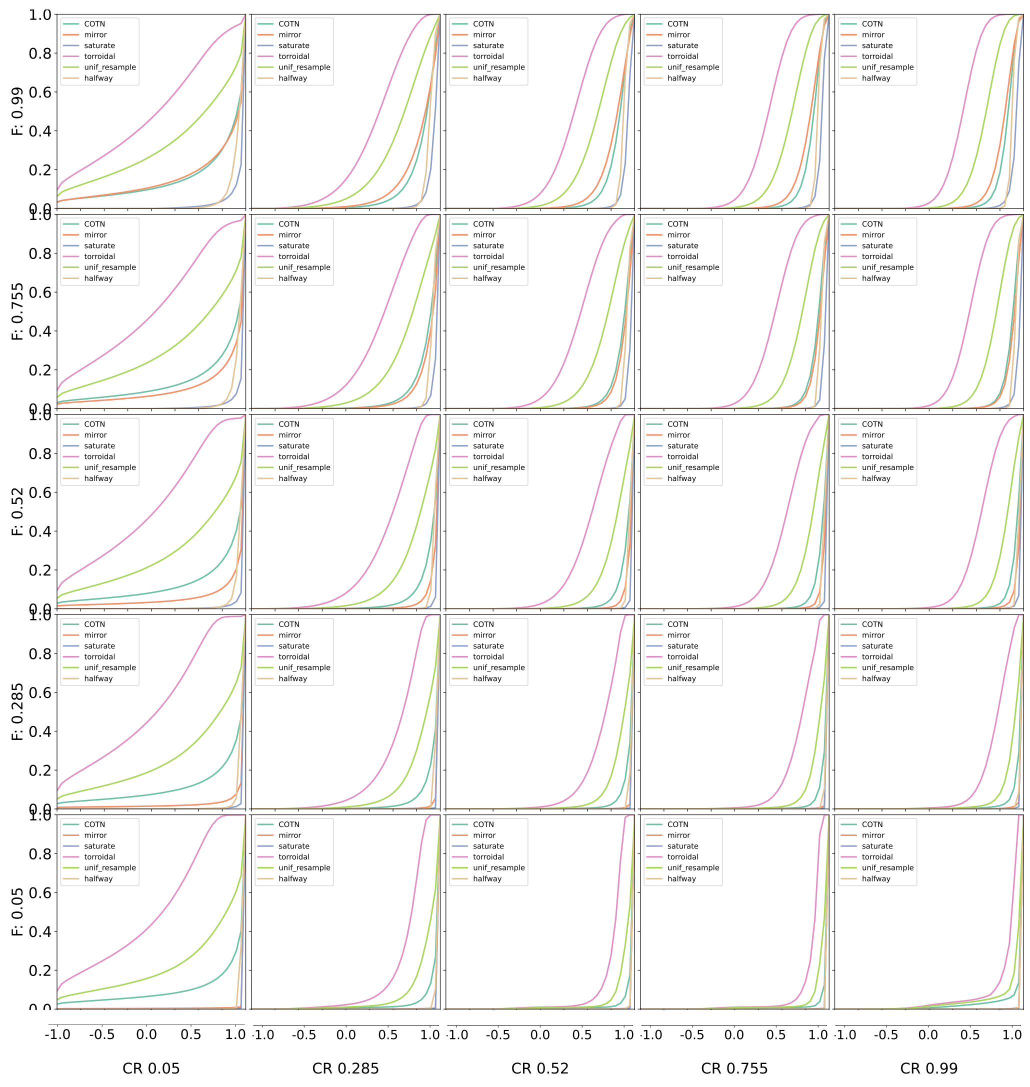


Figure 2: ECDF curves of the cosine similarity values on f_0 for `DE/rand/1/bin-p100` for different SDIS and values of F (subfigures in vertical direction) and C_r (subfigures in horizontal direction), 30 runs each.

the same quadrant); (iii) the cosine similarity between the initial search direction and that corresponding to `sat` is always positive and higher than that corresponding to `HVB` (Mitran, 2021).

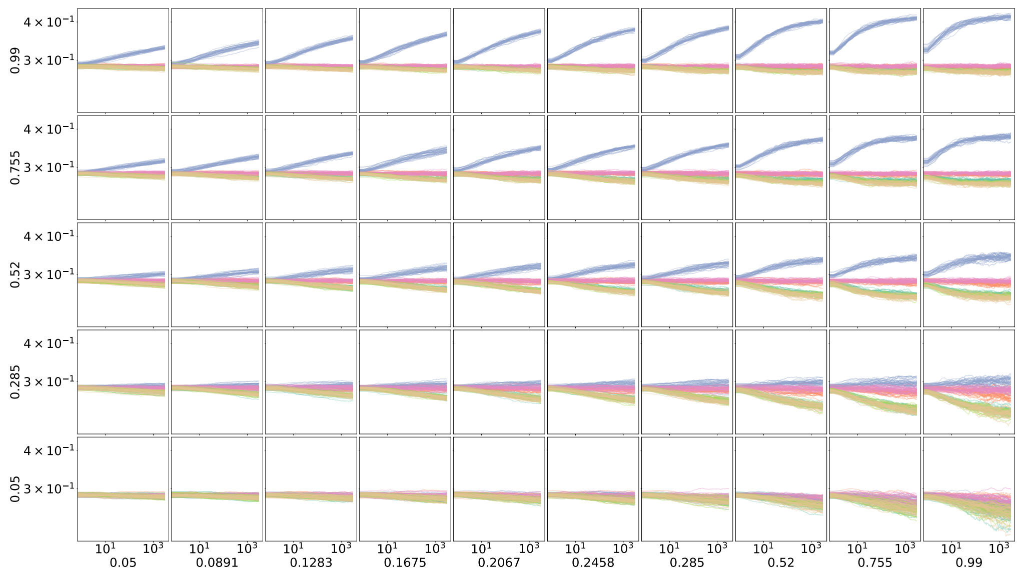


Figure 3: Evolution of population diversity (computed as explained in Section 5.3) per generation on f_0 for 30 runs of `DE/rand/1/bin-p100` for different SDIS, values of F (subfigures in vertical direction), C_r^{bin} (subfigures in horizontal direction). Colours define SDIS variant and correspond to those in Figure 2.

5.3 Implications of direction-disruptive SDIS operators on population diversity

To further measure the impact of SDIS, we track *population diversity* on f_0 in each generation g for multiple runs. The resulting trends are shown in Figure 3, where the diversity of the population is the averaged standard deviation of the individuals that form the g^{th} generation calculated per dimension ($n = 30$), after the application of different SDIS.

Following the notation introduced in Section 4.4 to obtain the variances, this value is simply calculated as $\text{std}^{(g)}(X) = \frac{1}{n}\sum_{i=1}^{n}\sqrt{\text{var}^{(g)}(X_i)}$. For the sake of clarity, note that the theoretical analysis was performed, for computational reasons, in terms of variance, while the graphical illustration of the evolution of population diversity is based on the standard deviation.

Figure 3 demonstrates a rather consistent picture for `DE/rand/1/bin-p100` in f_0 for all values of F excluding the smallest and for all values of C_r: (i) diversity induced by `sat` variant shown in blue is consistently the highest and increases during the runs;

(ii) the second highest diversity values are attained consistently by the variants tor and mir shown in pink and orange, respectively; these two variants are nearly constant during the runs and indistinguishable in terms of diversity; (iii) the third group of variants, also indistinguishable in terms of diversity, is demonstrated by COTN, uni and HVB; the diversity corresponding to these variants is consistently decreasing during the runs. Meanwhile, for the smallest value of $F = 0.05$, the diversity of all variants is largely indistinguishable and decreases over time. In general, the increase and decrease of diversity during the run depend on both DE control parameters, however, the increase in diversity in the sat variant is more drastic than the decrease of variants from the third group discussed above.

Furthermore, comparing the results of disruptiveness (Section 5.2) with those related to diversity, it turns out that a more disruptive strategy *does not* necessarily induces an increase in population diversity (when diversity is quantified using population variance).

Relation to theoretical results. The theoretical results on population diversity presented in Section 4.4 (for sat, uni and mir) state that the variance of the population of *corrected components* is the largest in the case of sat and the smallest in the case of uni, being equal to $(b-a)^2/4$ and $(b-a)^2/12$, respectively. The variance of the population of components corrected by mir decreases with F from the value corresponding to sat to that corresponding to

texttttCOTN which is slightly larger than that of components corrected by uni (Fig. 13, Appendix A.4). On the other hand, as also stated in Section 4.4, the symmetry between mir and tor leads to the same variance of the population of components corrected by these two SDISs. Thus, as illustrated in Fig. 14 (Appendix A.4) the experimental results are *consistent* with what is expected from the theoretical point of view.

The *difference in the shape* of the standard deviation curves as illustrated in Figures 3 and 14 (Appendix A.4) can be explained by the fact that the theoretical curves have been estimated in the case of a flat function, i.e., all new elements are incorporated into the population after correction, while for f_0 acceptance is based on a random decision (via the value of the objective function), so the change in the populations corresponding to

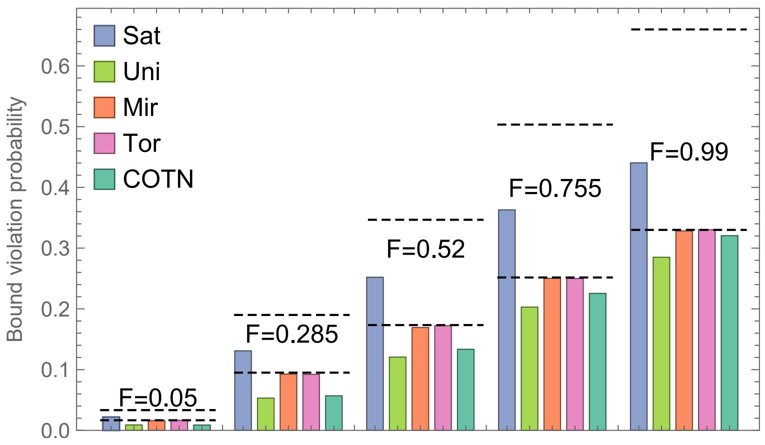


Figure 4: Empirical probability of bounds violation (averages of the relative frequencies of infeasible components) in `DE/rand/1/⋆-p100` on f_0 of dimensionality $n = 30$, estimated over 100 generations and 30 independent runs. The dashed lines correspond to the lower and upper bounds inferred for `sat` under the assumptions stated in Appendix A.1: $F/3$ and $2F/3$, respectively).

two consecutive generations is expected to be smaller than in the case of a flat function. This might explain the more gradual change in the diversity measure in the case of empirical analysis (Fig. 3) than in the case of theoretical estimations (Fig. 14 in the Appendix A.4) where the limit values are reached rather quickly.

5.4 Relation to the analysis of bound violation probabilities

To collect information on the influence of SDIS on the number of infeasible solutions, an experimental analysis was carried out on f_0 using a population of $N = 100$ elements and collecting the number of infeasible components during the first 100 generations of `DE/rand/1/⋆`, in the case of the 5 values of F, as used in previous experiments. The probability of bound violation has been estimated as the average frequency of infeasible components and the estimations for five SDISs are illustrated in Figure 4.

The main remark is that `sat` generates the highest number of infeasible components followed by `mir` and `tor` which are characterised by the same bound violation probability. The smallest number of infeasible components seems to be generated by `COTN` and by `uni`.

Relation to theoretical results.

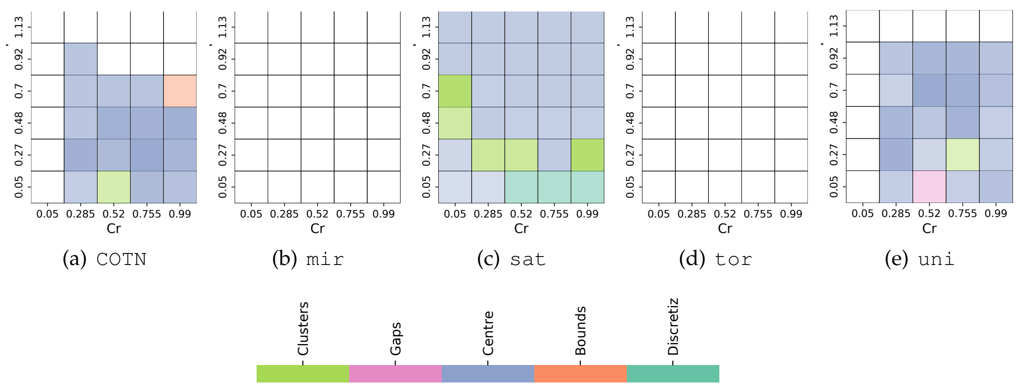
In the absence of selection-induced bias, it is expected that the probability of generating, by `DE/rand/1` mutation, components that are outside the bounds is close to that estimated theoretically, that is, $p_v = F/3$. However, the inclusion of corrected components in the population, during the evolutionary generations, can lead to changes in the probability of violation of the bounds, as inferred in Proposition 1 (Appendix A.1) for `sat`. As is illustrated in Fig. 4, the empirically estimated bound violation probability for `sat` is between the limits established for `sat` under the assumptions stated in Appendix A.1, i.e. $F/3$ and $2F/3$.

5.5 Relation to the analysis of structural bias

The test function f_0 used in this investigation was originally introduced (Kononova et al., 2015) to study the so-called structural bias (SB) of a heuristic optimisation algorithm, which is an inability of an algorithm to explore different parts of the search domain to the same extent regardless of the objective function. Such a study requires decoupling the effects of the landscape of the objective function from that of SB. It is precisely the random nature of f_0 and thus the known distribution of locations of its optima in a series of independent runs that allows the identification of SB: an algorithm is said to suffer from SB if the locations of the final best points found in a series (of a reasonable size) of independent runs minimising f_0 produced within a realistic budget of fitness evaluations deviate from uniform (Kononova et al., 2015).

As the nature of SB appears to originate from the iterative application of a limited set of algorithm operators, its identification is not straightforward (Kononova et al., 2020b,a; Vermetten et al., 2022c). Among the various algorithms investigated in literature so far, the results on SB in DE show clear patterns in time (van Stein et al., 2021), dimensionality (Vermetten et al., 2021) and parameter space (Vermetten et al., 2022d). Referring to the latter, Figure 5 shows such results on the presence and type of SB identified for `DE/rand/1/bin-p100` for various values of F and C_r with 5 variants of SDIS considered in this paper.

These results support the picture in Figure 3: (i) `mir` and `tor` indeed stay constant



(a) `COTN` (b) `mir` (c) `sat` (d) `tor` (e) `uni`

Figure 5: Predictions on the kind of structural bias (with confidences as colour intensities) produced by the random forest model from the `BIAS` toolbox (Vermetten et al., 2022c) for `DE/rand/1/bin-p100` with 5 SDIS variants, $F \in [0.05, 1.13]$ and $C_r \in [0.05, 0.99]$. White squares correspond to configurations with no structural bias detected. The figure is taken from (Vermetten et al., 2022d), which has not considered the `HVB` SDIS.

in terms of diversity, which is what we would expect from an unbiased algorithm; (ii) the 'most' biased variant (`sat`) also has the largest difference in terms of diversity, while `uni` and `COTN` are somewhat in-between.

Relation to theoretical results. The absence of structural bias in the case of `mir` and `tor` suggests that the assumption of uniformly distributed populations, used in the theoretical analysis, is not too restrictive, at least for these strategies. This aspect is illustrated, for `mir` and `tor`, through the closeness between the theoretically and empirically estimated values of the bound violation probability (Figure 4). On the other hand, in the case of `sat`, `uni`, and `COTN`, the presence of structural bias induces a deviation from the uniformity assumption, and consequently, the empirical violation probabilities are not so close to the theoretical values obtained based on the uniformity assumption.

6 Experimentation on the BBOB suite

To assess whether the observed differences in SDIS have an impact on DE performance outside of f_0, we conducted a benchmarking study on COCO benchmarking suite (Hansen et al., 2021). We make use of the single-objective, noiseless version of the BBOB function set, which contains 24 distinct functions (Finck et al., 2010), each of which can be instantiated with different transformations, referred to as instances of these functions. The BBOB-functions are all defined using box-constraints of $[-5, 5]^n$, where n is the dimension of the problem, and are accessed here using IOHexperimenter (de Nobel et al., 2021).

Where possible, results on BBOB functions discussed in this section are *contrasted* with theoretical conclusions obtained under a number of simplifying assumptions made in Section 4.

6.1 Setup for the experimentation on the BBOB suite

To run Differential Evolution algorithms on the BBOB-functions, we make use of a python-based implementation that incorporates several variants of DE, including `SHADE` and `L-SHADE` employed for this study – see Section 2.2 for their description.

The code for both this version of Differential Evolution and the complete benchmarking setup on BBOB can be found on GitHub [19]. Furthermore, all reproducibility steps can be found on the same GitHub page. For the BBOB-based experiments described in this section, we use both the 5- and 30-dimensional versions of the problems, using instances 1-5, and perform 5 runs per instance. We use a total budget of $10000 \cdot n$ fitness evaluations.

6.2 Analysis of performance benchmarking

The most common way to analyse the performance of an iterative optimisation heuristic is by considering *Expected Running Time (ERT)*, defined as follows.

Given an algorithm A, a function instance F, a target ϕ and the run number i, we define the hit time $t_i(A, F, \phi)$ as the number of function evaluations that the algorithm used in its i^{th} run before evaluating a quality solution of at least ϕ. Based on this, we can then define the expected running time in a set of I runs and J problem instances as follows:

$$\mathrm{ERT}(v, \mathcal{F}, \phi) = \frac{\sum_{i=1}^{I} \sum_{j=1}^{J} \min\{t_i(v, f^{(j)}, \phi), B\}}{\sum_{i=1}^{I} \sum_{j=1}^{J} \mathbb{1}\{t_i(v, f^{(j)}, \phi) < \infty\}}$$

[19] `https://github.com/Dvermetten/DE_TIOBR`

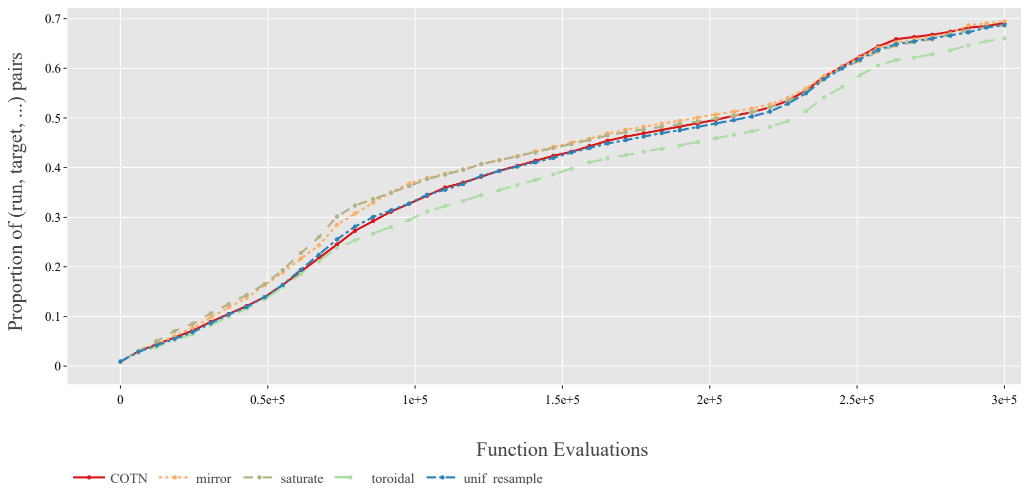


Figure 6: ECDF curves of L-SHADE with 5 different SDIS, aggregated across all 24 30-dimensional BBOB functions with 5 instances and 5 runs each, using 51 logarithmically spaced fitness targets between 10^2 and 10^{-8}.

where B denotes the computational budget with respect to the number of function evaluations.

In Figure 7, we show the ERT (fraction of runs that reach the specified targets in their respective function vs the number of function evaluations) achieved by the L-SHADE algorithm (see Section 2.2) with 5 different SDIS variants on the 30-dimensional BBOB functions, per function.

From this figure, we can clearly see that for many functions, the effect of SDIS on overall performance is relatively minor. However, there are some outliers, in particular f_5 (linear slope). Since for f_5 the optimum is located on the bound in each coordinate, the SDIS method will have a significant impact on the ability of DE to converge. When zooming in on this function, we clearly see that sat outperforms all other SDIS, followed by mir, and then the other SDISs. This performance ordering is also present for some other unimodal functions such as f_1 (Sphere) and f_2 (Ellipsoidal).

When aggregating the performance across all functions, which we can do using the ECDF as seen in Figure 6, we actually observe clear differences in overall performance between the different L-SHADE versions. Although the ECDF is obviously affected by the linear slope function, even when this is removed from consideration, the ordering

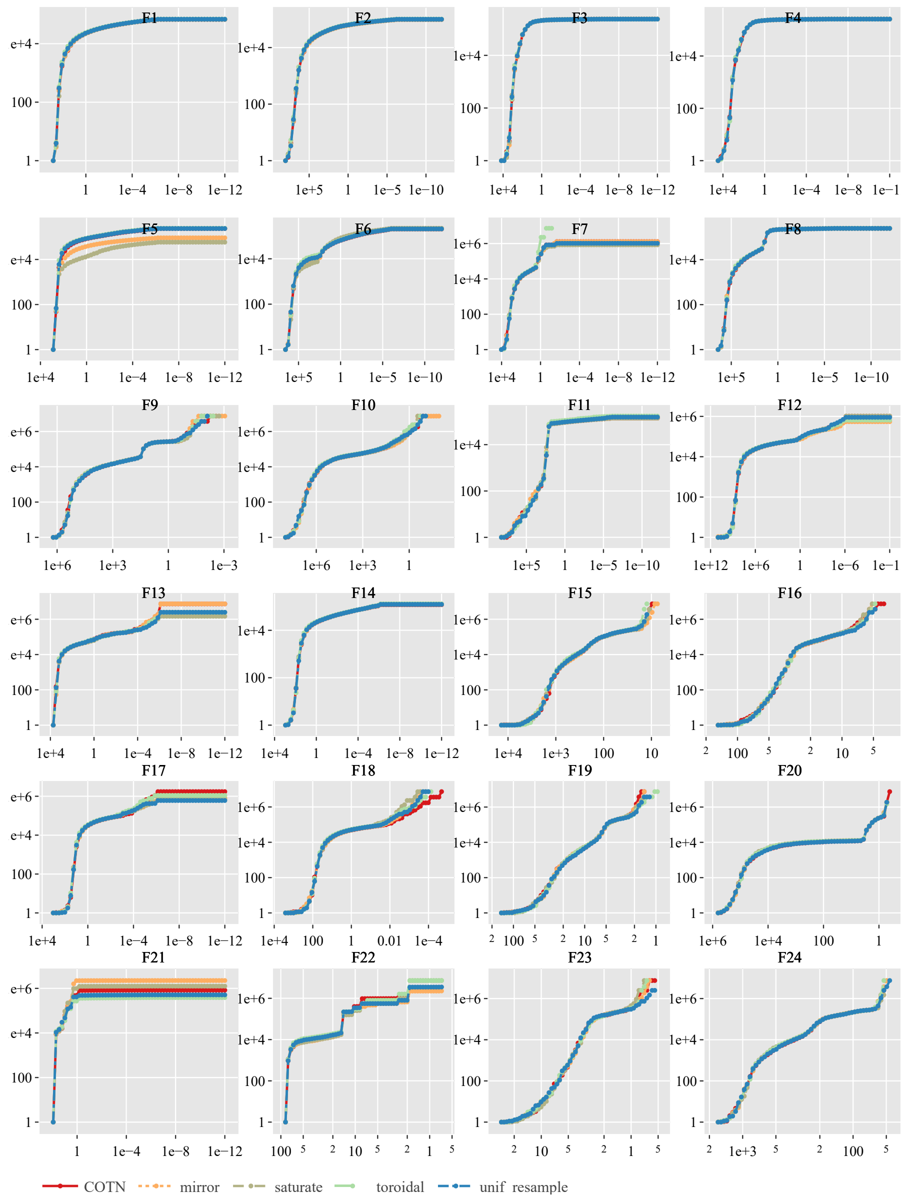


Figure 7: Overview of the ERT (shown in the vertical axis) vs best-so-far $f(x)$ (shown on the horizontal axis) for `L-SHADE` with 5 SDIS methods on the BBOB-functions in 30D, 5 instances 5 runs each.

of SDIS-variants remains the same and matches the order of least disruptiveness in the search, as discussed in Section 5.2. In addition to running `L-SHADE` on the 30-dimensional BBOB-functions, we have also collected data on the 5D version for both

L-SHADE and SHADE. All figures are available on figshare (Vermetten et al., 2022b), while the extended data set is available directly on the IOHanalyzer GUI [20].

Overall, we can see that the choice of SDIS indeed has an impact on the final performance of the algorithm, and although this impact is not present on all functions, it is a clear indication that SDIS should be considered a part of the specification of the algorithm. Furthermore, the fact that the differences are not equally present on all functions indicates that methods for per-instance algorithm configuration could benefit from including the SDIS in their search space.

6.3 Cosine similarity distributions on BBOB

The distributions of the cosine similarity values corresponding to five variants of SDIS (COTN, mir, sat, tor, and uni) have been generated for 24 BBOB functions both when L-SHADE is used (Figure 8) and when SHADE is used - omitted for space limitation; see the extensive set of graphical results in (Vermetten et al., 2022b).

The behaviour patterns are only slightly different between the two methods. When comparing the ECDFs illustrated in Figure 8 with those obtained by applying DE/rand/1/$\star$ on f_0 (Figure 2), we can see that the curves have shifted as a result of the different objective functions landscapes, but the global ordering is preserved almost everywhere with sat SDIS characterised by the largest CS values and tor by the smallest ones. The presence of different patterns for different functions supports the idea that SDIS should be considered a separate algorithmic component.

Relation to theoretical results. Our experimental results on BBOB match our theoretical insights. Although our proofs on the relation between mir and tor make some particular assumptions, the empirical analysis shows that mir indeed preserves more of the search direction than tor in all except one of the tested functions (f_5). The significance of differences in cosine values between different SDISs has been confirmed by running pairwise Kolmogorov-Smirnov tests with Benjamini-Yukateli correction $\alpha = 0.01$. The overall ordering of cosine similarities is consistent for almost all BBOB functions and matches that observed on f_0.

[20] Available as data set source 'TIOBR_DE' on https://iohanalyzer.liacs.nl

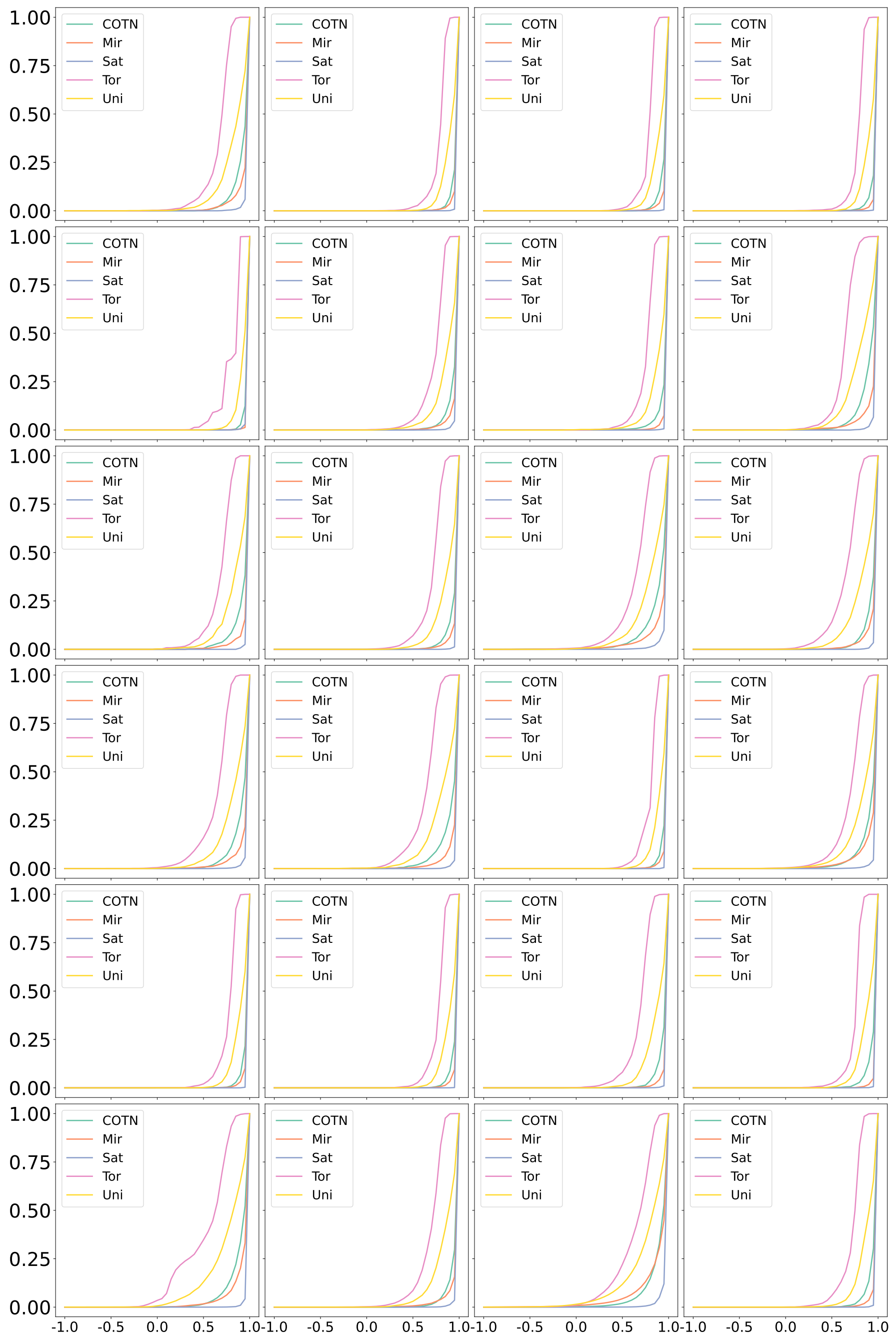
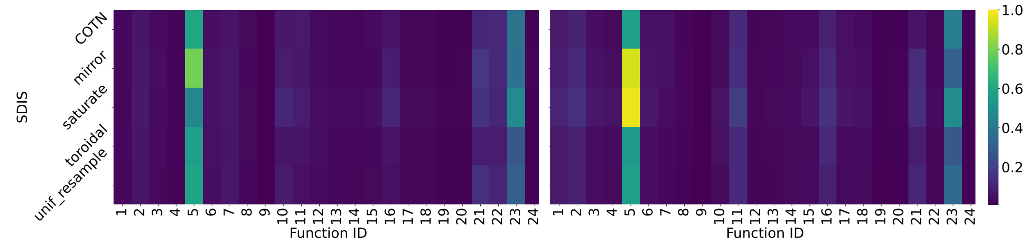


Figure 8: ECDF curves of cosine similarity values of corrected infeasible solutions generated during 5 independent full-budget runs of `L-SHADE` on 24 BBOB functions (all instance 1) for different SDIS variants (one subfigure per function, functions are shown left to right, top to bottom).



(a) 5D (b) 30D

Figure 9: Final percentages of infeasible solutions generated in full-budget runs of L-SHADE with different SDIS variants on 24 functions of the BBOB suite (instance 1 only) averaged over 5 runs per variant per function. Vertical labels apply to both plots.

6.4 Analysis of final percentage of infeasible solutions

In addition to looking at the performance of DE with different SDISs, we can also zoom in on the related behaviour of the algorithm. Since a SDIS is activated only when solutions outside of the bounds are generated, we can consider the Percentage Of Infeasible Solutions (POIS) generated throughout the optimisation process. The final POIS values (that is, the number of infeasible solutions generated at the end of the run to the total fitness evaluation budget) from the L-SHADE algorithm produced on the BBOB functions is shown in Figure 9, from which we can see that for the higher dimensionality ($n = 30$), the fraction of infeasible solutions generated during the search is slightly higher than for lower dimensionality ($n = 5$).

This can be explained by considering that we count a solution to be infeasible when one or more components are outside their boundaries, which is more likely to occur when there are more components in a solution (see also Section 2.3). However, since for the BBOB suite of problems the optimum is guaranteed to be at least 10% away from the bound on each side in all functions except the linear slope Finck et al. (2010), the effect of increased dimensionality is not as obvious in those functions.

A difference between SDISs is visible, particularly in the case of function landscapes that enforce the search nearby the boundary (e.g. f_5, f_{23}). For the other functions, infeasible individuals are generated mainly in the first generations, thus the aggregated POIS values are mainly influenced by the small number of infeasible individuals generated during most of the evolution.

We additionally see that the POIS varies greatly between functions, with the linear slope (f_5) clearly showing a large number of infeasible solutions as would be expected.

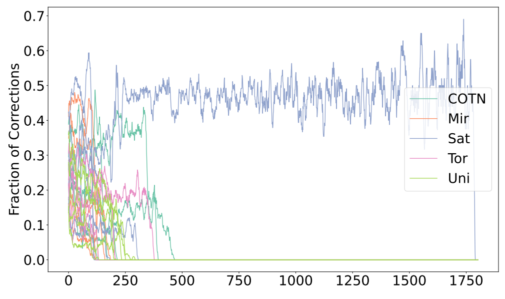
6.5 Analysis of windowed percentage of infeasible solutions and population diversity

While the final POIS discussed in Section 6.4 is useful in providing information on the total number of infeasible solutions, we can get a more detailed overview of the number of infeasible solutions generated using a sliding window approach: for each generation, we can calculate the fraction of infeasible solutions generated *thus far* [21] and visualise the moving average of 10 generations. We can use a similar approach for population diversity, which allows us to check for possible correlations between ongoing POIS and diversity. Figures 10 and 11 visually show these two types of graph for functions f_5 and f_{23}, respectively.

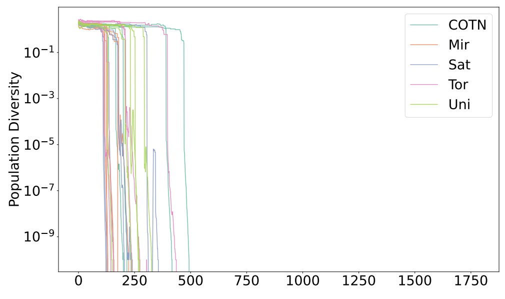
In the case of the multi-modal f_{23} function (Figure 10) the behaviour of `SHADE` is in accordance with some of the theoretical insights regarding the number of infeasible solutions (largest bound violation probability in the case of `sat`) and the evolution of population diversity (largest diversity in the case of `sat`, smallest in the case of `uni` and `COTN` and intermediate in the case of `mir` and `tor`). Meanwhile, in particular, for f_5, we can observe in Figure 11 a clear pattern between the different variants of SDIS: `sat` initially has a much larger POIS, which would be beneficial, since the optimum of this function is located on the bounds. The other SDISs do not have this direct benefit of `sat`, and need to generate a point on the boundary exactly, without the correction moving it away from the optimum. For the 5-dimensional version of the function, this is still achievable for most SDISs, with the lower disruptiveness of `mir` allowing it to find the optimum relatively easily, but for 30 dimensions, the most disruptive SDISs have runs that do not converge to a single solution, which explains their poor performance seen in Figure 7.

We also notice that the differences between SHADE and L-SHADE are rather significant, especially on the higher dimensions – detailed reasons for this require further

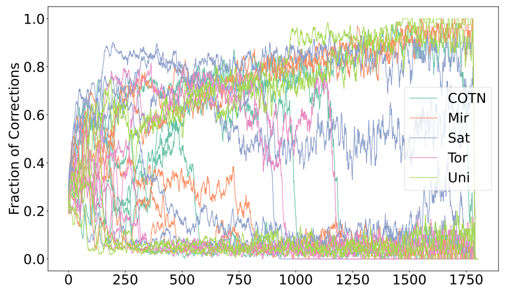
[21] Equivalent to the fraction of solutions which have had a SDIS applied.



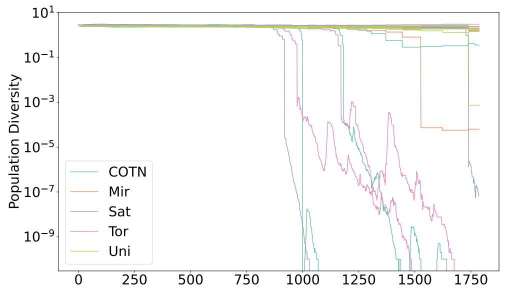
(a) Windowed POIS, `SHADE`, 5D



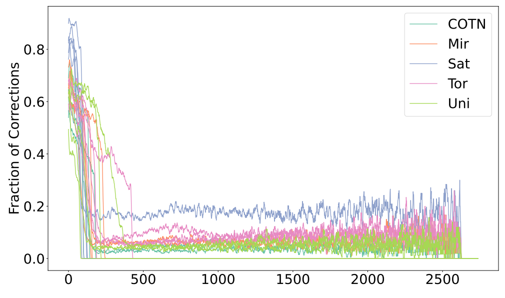
(b) Population diversity, `SHADE`, 5D



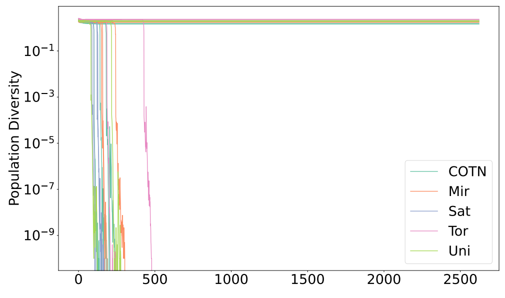
(c) Windowed POIS, `L-SHADE`, 5D



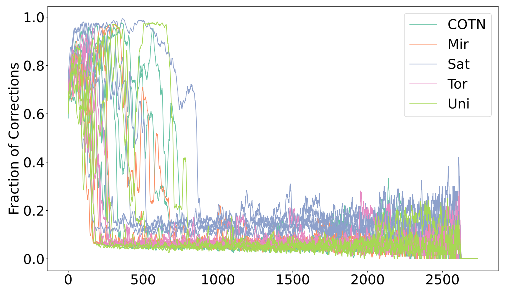
(d) Population diversity, `L-SHADE`, 5D



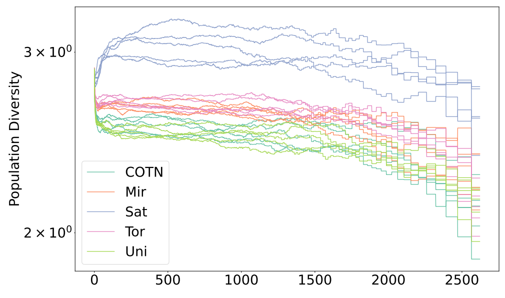
(e) Windowed POIS, `SHADE`, 30D



(f) Population diversity, `SHADE`, 30D

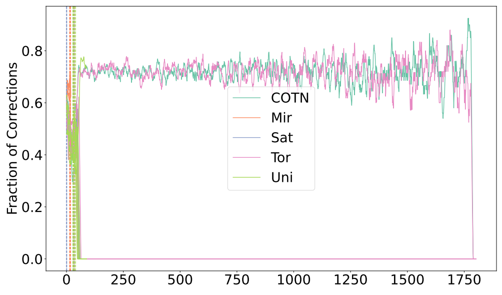


(g) Windowed POIS, `L-SHADE`, 30D

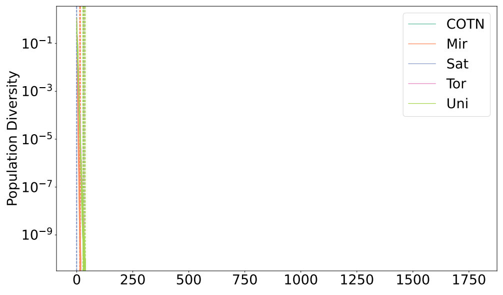


(h) Population diversity, `L-SHADE`, 30D

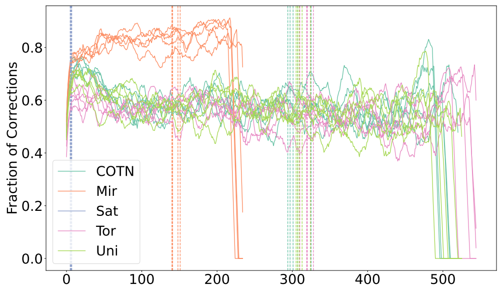
Figure 10: Evolution of windowed POIS (left column) and population diversity (right column) over generations for different SDIS methods on 5 runs of f_{23}, instance 1. None of the runs reached the optimum.



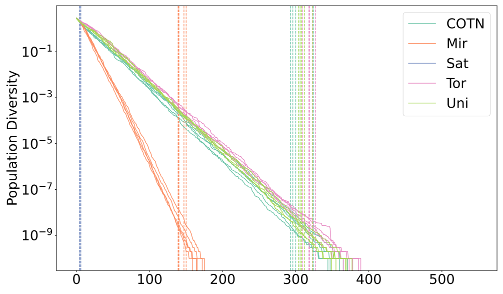
(a) Windowed POIS, `SHADE`, 5D



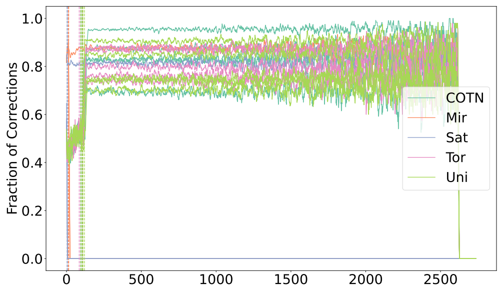
(b) Population diversity, `SHADE`, 5D



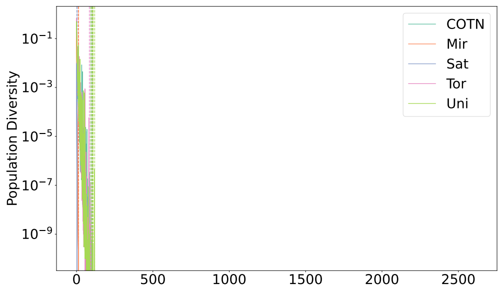
(c) Windowed POIS, `L-SHADE`, 5D



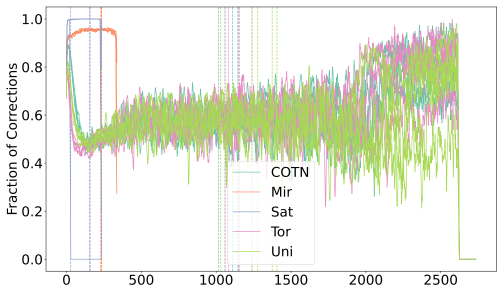
(d) Population diversity, `L-SHADE`, 5D



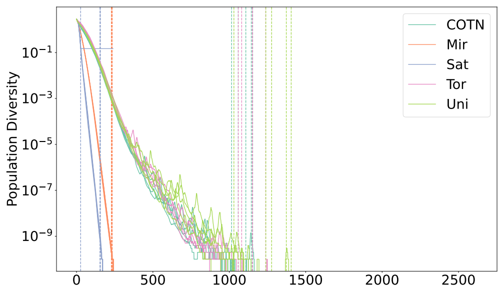
(e) Windowed POIS, `SHADE`, 30D



(f) Population diversity, `SHADE`, 30D



(g) Windowed POIS, `L-SHADE`, 30D



(h) Population diversity, `L-SHADE`, 30D

Figure 11: Evolution of windowed POIS (left column) and population diversity (right column) over generations for different SDIS methods on 5 runs of f_5, instance 1. Coloured vertical lines show times when the optimum was found at least once in a run.

Table 3: Ranking of SDIS based on theoretical (obtained where possible) and empirical results with respect to the numbe of infeasible components, search direction disruptiveness measured using cosine similarity, population diversity measured using the component-wise variance and performance measured using ERT.

Aspect	Number of infeasible components		Amount of disruptiveness		Increase of population diversity		Fitness-based performance
Sorting	smaller-larger		smaller-larger		larger-smaller		best-worst
Algorithm	`DE/rand/1/*`		`DE/rand/1/*`		`DE/rand/1/*`		`(L-)SHADE`
Type	theor.	empir.	theor.	empir.	theor.	empir.	empir.
Function(s)	flat	f_0	flat	f_0	flat	f_0	BBOB
Section	4.1	5.4	4.3	5.2	4.4	5.3	6.2
1.	`uni`	`uni`	`sat`	`sat`	`sat`	`sat`	`sat`
2.	`sat`	`COTN`	`mir`	`COTN` `mir`	`mir` `tor`	`mir` `tor`	`COTN` `mir`
3.		`mir` `tor`	`tor`	`uni`	`COTN` `uni`	`COTN` `uni`	`uni`
4.		`sat`		`tor`			`tor`

investigation.

7 Conclusions

The results of the comparative analysis on SDIS presented in this article are summarised in Table 3 where the investigated strategies are ranked according to what has been theoretically proved and/or experimentally observed.

Despite the fact that the theoretical analysis is limited to subsets of strategies, consistency can be observed between theoretical and experimental results on how various SDISs can be grouped based on their impact on: (i) number of infeasible components; (ii) search direction; (iii) population diversity. The most significant agreement appears to be between the amount of disruptiveness and fitness-based performance, suggesting that the *strategies with a smaller impact on the search direction*, i.e., higher cosine similarity between unconstrained and corrected search directions, *have a better performance*. When compared with previously reported results, one can see that the more disruptive strategies, `uni` and `tor`, have also been identified in (Biedrzycki et al., 2019) as having the highest influence on population distribution. Similarly, in (Padhye et al., 2015) it is stated that `uni` leads to a loss of useful information carried by the current population which confirms the disruptive effect also observed in the current study. On the other hand, `sat` and `mir` identified in our study with a small impact on search direction but a large impact on diversity have been consistently among well-performing strategies, as is also illustrated in (Kreischer et al., 2017) and (de-la-Cruz-Martínez and Mezura-Montes, 2020). However, the performance advantage of the less disruptive strategies in BBOB is not present in all functions equally.

Throughout this paper, we have shown that the strategy of dealing with infeasible solutions within Differential Evolution has a clear impact on the behaviour of the algorithm as a whole. Although DE allows us to combine insights on the impact of SDIS from both a theoretical and empirical perspective, it is by no means the only algorithm where the way of handling bound constraints can have an impact on overall search behaviour. This highlights an important issue in the field of evolutionary computation as a whole, because methods such as SDIS are often overlooked because they are considered unimportant compared to the novel algorithmic ideas discussed in the literature. When this omission is combined with other factors, such as the inaccessibility of the source code used, this leads to a significant amount of ambiguity, even when other operators are defined clearly. As such, reproducing results requires specificity and, ideally, open source code for *all* the operators used, not only the core algorithmic components.

To achieve *better standard for reproducibility*, SDIS should be considered as an operator to be specified in every optimisation algorithm which deals with bound constraints, as for some functions the interaction of the algorithm with bounds can cause clear differences in behaviour, especially for some of the high-dimension version of functions in the BBOB-set.

Finally, as such, we see potential benefits for the inclusion of SDIS in *automatic algorithm configuration task*.

8 Future work

Although we have investigated the impact of SDIS on several variants of differential evolution throughout this work, it remains formally unconfirmed that such findings translate to other optimisation algorithms. Since many of these algorithms are built to work with bound constraints by default, it stands to reason that the SDIS used would have an influence, and by studying this in more detail, we could potentially broaden our understanding of the interactions between SDIS, algorithm, and objective function.

Furthermore, we have started to lay a theoretical foundation for the study of several SDISs by considering their disruptiveness and impact on diversity. By continuing to build upon these notions, we hope to gain a more detailed understanding of the impact of SDIS on search behaviour, in general.

References

Ali, M. and Fatti, L. (2006). A differential free point generation scheme in the differential evolution algorithm. *Journal of Global Optimization*, 35:551–572.

Arabas, J., Szczepankiewicz, A., and Wroniak, T. (2010). Experimental comparison of methods to handle boundary constraints in differential evolution. In *Parallel Problem Solving from Nature, PPSN XI*, volume 6239 of *Lecture Notes in Computer Science*, pages 411–420. Springer.

Awad, N., Ali, M., Mallipeddi, R., and Suganthan, P. (2018). An improved differential evolution algorithm using efficient adapted surrogate model for numerical optimization. *Information Sciences*, 451.

Biedrzycki, R. (2019). Handling bound constraints in CMA-ES: an experimental study. *Swarm and Evolutionary Computation*, 52:100627.

Biedrzycki, R., Arabas, J., and Jagodziński, D. (2019). Bound constraints handling in differential evolution: An experimental study. *Swarm and Evolutionary Computation*, 50:100453.

Boks, R. (2021). Dynamic configuration of operators and parameters in differential evolution through combined fitness and diversity-driven adaptation methods. Master's thesis, LIACS, Leiden University.

Boks, R., Kononova, A. V., and Wang, H. (2021). Quantifying the impact of boundary constraint handling methods on differential evolution. In *Proceedings of the 2021 Genetic and Evolutionary Computation Conference Companion*, GECCO '21 Companion, pages 1199—1207.

Brest, J., Maučec, M. S., and Bošković, B. (2019). The 100-digit challenge: Algorithm jde100. In *2019 IEEE Congress on Evolutionary Computation (CEC)*.

Bujok, P., Tvrdík, J., and Poláková, R. (2014). Differential evolution with rotation-invariant mutation and competing-strategies adaptation. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 2253–2258.

Caraffini, F. and Iacca, G. (2020). The SOS platform: Designing, tuning and statistically benchmarking optimisation algorithms. *Mathematics*, 8(5):785.

Caraffini, F. and Kononova, A. V. (2019). Structural bias in differential evolution: a preliminary study. *AIP Conference Proceedings*, 2070(1):020005.

Caraffini, F., Kononova, A. V., and Corne, D. W. (2019). Infeasibility and structural bias in differential evolution. *Information Sciences*, 496:161–179.

Caraffini, F. and Neri, F. (2019). A study on rotation invariance in differential evolution. *Swarm and Evolutionary Computation*, 50:100436.

Cheng, J., Pan, Z., Liang, H., Gao, Z., and Gao, J. (2021). Differential evolution algorithm with fitness and diversity ranking-based mutation operator. *Swarm and Evolutionary Computation*, 61:100816.

Cheng, L., Wang, Y., Wang, C., Mohamed, A. W., and Xiao, T. (2020). Adaptive differential evolution based on successful experience information. *IEEE Access*, 8:164611–164636.

Cheng, S., Shi, Y., and Qin, Q. (2011). Experimental study on boundary constraints handling in particle swarm optimization: From population diversity perspective. *International Journal of Swarm Intelligence Research*, 2(3):43–69.

Choi, T. J., Togelius, J., and Cheong, Y.-G. (2020). Advanced cauchy mutation for differential evolution in numerical optimization. *IEEE Access*, pages 8720–8734.

Cleghorn, C. W. and Engelbrecht, A. P. (2014). Particle swarm convergence: An empirical investigation. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 2524–2530.

Coello Coello, C. A. (2002). Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, 191(11):1245–1287.

Cruz-Duarte, J. M., Ortiz-Bayliss, J. C., Amaya, I., Shi, Y., Terashima-Marín, H., and Pillay, N. (2020). Towards a generalised metaheuristic model for continuous optimisation problems. *Mathematics*, 8(11).

Das, S., Mullick, S. S., and Suganthan, P. (2016). Recent advances in differential evolution – an updated survey. *Swarm and Evolutionary Computation*, 27:1 – 30.

de-la-Cruz-Martínez, S. and Mezura-Montes, E. (2020). Boundary constraint-handling methods in differential evolution for mechanical design optimization. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE.

de Nobel, J., Ye, F., Vermetten, D., Wang, H., Doerr, C., and Bäck, T. (2021). IOHexperimenter: Benchmarking platform for iterative optimization heuristics. *CoRR*, abs/2111.04077.

Deb, K. (2000). An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(2):311–338.

Deng, L., Li, C., Lan, Y., Sun, G., and Shang, C. (2022). Differential evolution with dynamic combination based mutation operator and two-level parameter adaptation strategy. *Expert Systems with Applications*, 192:116298.

Deng, W. (2020). Differential evolution algorithm with wavelet basis function and optimal mutation strategy for complex optimization problem. *Applied Soft Computing*, 100.

Engelbrecht, A. (2013a). Fruitless search in differential evolution. In *2013 IEEE Symposium on Differential Evolution (SDE)*, pages 9–17.

Engelbrecht, A. (2013b). Roaming behavior of unconstrained particles. In *2013 BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence*, pages 104–111.

Finck, S., Hansen, N., Ros, R., and Auger, A. (2010). Real-parameter black-box optimization benchmarking 2009: Presentation of the noiseless functions. Technical report, Citeseer.

Hansen, N., Auger, A., Ros, R., Mersmann, O., Tušar, T., and Brockhoff, D. (2021). COCO: A platform for comparing continuous optimizers in a black-box setting. *Optimization Methods and Software*, 36(1):114–144.

Helwig, S., Branke, J., and Mostaghim, S. (2013). Experimental analysis of bound handling techniques in particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 17(2):259–271.

Helwig, S. and Wanka, R. (2008). Theoretical analysis of initial particle swarm behavior. In Rudolph, G., Jansen, T., Beume, N., Lucas, S., and Poloni, C., editors, *Parallel Problem Solving from Nature – PPSN X*, pages 889–898, Berlin, Heidelberg. Springer Berlin Heidelberg.

Juárez-Castillo, E., Acosta-Mesa, H., and Mezura-Montes, E. (2017). Empirical study of bound constraint-handling methods in particle swarm optimization for constrained search spaces. In *2017 IEEE Congress on Evolutionary Computation*, pages 604–611. IEEE.

Kadavy, T., Viktorin, A., Kazikova, A., Pluhacek, M., and Senkerik, R. (2022). Impact of boundary control methods on bound-constrained optimization benchmarking. *IEEE Transactions on Evolutionary Computation*, 26(6):1271–1280.

Kononova, A. V., Caraffini, F., and Bäck, T. (2021). Differential evolution outside the box. *Information Sciences*, 581:587–604.

Kononova, A. V., Caraffini, F., Wang, H., and Bäck, T. (2020a). Can compact optimisation algorithms be structurally biased? In *Parallel Problem Solving from Nature – PPSN XVI*, pages 229–242, Cham. Springer International Publishing.

Kononova, A. V., Caraffini, F., Wang, H., and Bäck, T. (2020b). Can single solution optimisation methods be structurally biased? In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–9, Glasgow. IEEE.

Kononova, A. V., Corne, D. W., Wilde, P. D., Shneer, V., and Caraffini, F. (2015). Structural bias in population-based algorithms. *Information Sciences*, 298:468–490.

Kreischer, V., Magalhães, T., Barbosa, H., and Krempser, E. (2017). Evaluation of bound constraints handling methods in differential evolution using the CEC2017 benchmark. In *Anais do 13 Congresso Brasileiro de Inteligência Computacional*, pages 1–12. ABRICOM.

Kumar, A., Biswas, P. P., and Suganthan, P. N. (2022). Differential evolution with orthogonal array-based initialization and a novel selection strategy. *Swarm and Evolutionary Computation*, 68:101010.

Lampinen, J. and Zelinka, I. (2000). On stagnation of the differential evolution algorithm. In *Proceedings of* 6^{th} *International Mendel Conference on Soft Computing*, pages 76–83.

L'Ecuyer, P. and Simard, R. (2007). TestU01: A C library for empirical testing of random number generators. *ACM Transactions on Mathematical Software*, 33(4):1–40.

Liao, T., Molina, D., de Oca, M. A. M., and Stützle, T. (2014). A note on bound constraints handling for the ieee cec'05 benchmark function suite. *Evolutionary Computation*, 22(2):351–359.

Liu, X.-f., Zhan, Z.-H., Lin, Y., Chen, W.-N., Gong, Y.-J., Gu, T.-L., Yuan, H.-Q., and Zhang, J. (2019). Historical and heuristic-based adaptive differential evolution. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(12):2623–2635.

López-Ibáñez, M., Branke, J., and Paquete, L. (2021). Reproducibility in evolutionary computation. *ACM Transactions on Evolutionary Learning and Optimization*, 1(4):1–21.

Meng, Z., Chen, Y., and Li, X. (2020). Enhancing differential evolution with novel parameter control. *IEEE Access*, 8:51145–51167.

Mitran, M.-A. (2021). Analysis of the influence of bound constraint handling strategies on the search direction in differential evolution algorithms. In *2021 23rd International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pages 291–298.

Mohamed, A. W. (2018). A novel differential evolution algorithm for solving constrained engineering optimization problems. *Journal of Intelligent Manufacturing*, 29.

Mohamed, A. W., Hadi, A. A., and Mohamed, A. K. (2021). Differential evolution mutations: Taxonomy, comparison and convergence analysis. *IEEE Access*, 9:68629–68662.

Mohamed, A. W. and Mohamed, A. K. (2019). Adaptive guided differential evolution algorithm with novel mutation for numerical optimization. *International Journal of Machine Learning and Cybernetics*, 10(2):253–277.

Mousavirad, S. J. and Rahnamayan, S. (2020). A novel center-based differential evolution algorithm. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8.

Mousavirad, S. J., Schaefer, G., Korovin, I., Moghadam, M. H., Saadatmand, M., and Pedram, M. (2021). An enhanced differential evolution algorithm using a novel clustering-based mutation operator. In *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 176–181.

Oldewage, E. T., Engelbrecht, A. P., and Cleghorn, C. W. (2018). Boundary constraint handling techniques for particle swarm optimization in high dimensional problem spaces. In *Swarm Intelligence - 11th International Conference*, volume 11172 of *Lecture Notes in Computer Science*, pages 333–341.

Opara, K. R. and Arabas, J. (2019). Differential evolution: A survey of theoretical analyses. *Swarm and Evolutionary Computation*, 44:546–558.

Padhye, N., Mittal, P., and Deb, K. (2015). Feasibility preserving constraint-handling strategies for real parameter evolutionary optimization. *Computational Optimization and Applications*, 62:851–890.

Price, K., Storn, R. M., and Lampinen, J. A. (2006). *Differential evolution: a practical approach to global optimization*. Springer Science & Business Media.

Prügel-Bennett, A. (2010). Benefits of a population: Five mechanisms that advantage population-based algorithms. *IEEE Transactions on Evolutionary Computation*, 14(4):500–517.

Song, E. and Li, H. (2021). A self-adaptive differential evolution algorithm using oppositional solutions and elitist sharing. *IEEE Access*, 9:20035–20050.

Stanovov, V., Akhmedova, S., and Semenkin, E. (2020). Archive update strategy influences differential evolution performance. In *Advances in Swarm Intelligence*, pages 397–404. Springer International Publishing.

Storn, R. (1996). On the usage of differential evolution for function optimization. In *Proceedings of North American Fuzzy Information Processing*, pages 519–523. IEEE.

Storn, R. and Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359.

Stützle, T. and López-Ibáñez, M. (2019). *Automated Design of Metaheuristic Algorithms*, volume 272 of *International Series in Operations Research & Management Science*, page 127–146. Springer, Cham.

Sun, G., Xu, G., and Jiang, N. (2020). A simple differential evolution with time-varying strategy for continuous optimization. *Soft Computing*, 24(4):2727–2747.

Tanabe, R. and Fukunaga, A. (2013). Success-history based parameter adaptation for differential evolution. In *2013 IEEE Congress on Evolutionary Computation*, pages 71–78.

Tanabe, R. and Fukunaga, A. S. (2014). Improving the search performance of shade using linear population size reduction. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 1658–1665. IEEE.

Tian, M. and Gao, X. (2019). An improved differential evolution with information intercrossing and sharing mechanism for numerical optimization. *Swarm and Evolutionary Computation*, 50:100341.

Tomczak, J. M., Wundefinedglarz-Tomczak, E., and Eiben, A. E. (2020). Differential evolution with reversible linear transformations. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, page 205–206.

van den Honert, N., Vermetten, D., and Kononova, A. V. (2021). Benchmarking the status of default pseudorandom number generators in common programming languages. *CoRR*, abs/2109.12997.

van Stein, B., Caraffini, F., and Kononova, A. V. (2021). Emergence of structural bias in differential evolution. In *Proceedings of the 2021 Genetic and Evolutionary Computation Conference Companion*, GECCO '21 Companion.

Vermetten, D., Kononova, A. V., Caraffini, F., Mitran, M., and Zaharie, D. (2022a). The importance of being constrained - dataset. `www.doi.org/10.5281/zenodo.7115488`.

Vermetten, D., Kononova, A. V., Caraffini, F., Mitran, M., and Zaharie, D. (2022b). The importance of being constrained - figures. `www.doi.org/10.6084/m9.figshare.18319394.v2`.

Vermetten, D., Kononova, A. V., Caraffini, F., Wang, H., and Bäck, T. (2021). Is there anisotropy in structural bias? In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '21, page 1243–1250, New York, NY, USA.

Vermetten, D., van Stein, B., Caraffini, F., Minku, L. L., and Kononova, A. V. (2022c). Bias: A toolbox for benchmarking structural bias in the continuous domain. *IEEE Transactions on Evolutionary Computation*, 26(6):1380–1393.

Vermetten, D., van Stein, B., Kononova, A. V., and Caraffini, F. (2022d). Analysis of structural bias in differential evolution configurations. In *Differential Evolution: From Theory to Practice*, pages 1–22. Springer Singapore.

Viktorin, A., Senkerik, R., Pluhacek, M., Kadavy, T., and Zamuda, A. (2019). Dish algorithm solving the cec 2019 100-digit challenge. In *2019 IEEE Congress on Evolutionary Computation (CEC)*. IEEE Press.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272.

Wang, Z.-J., Zhan, Z.-H., Lin, Y., Yu, W.-J., Wang, H., Kwong, S., and Zhang, J. (2020). Automatic niching differential evolution with contour prediction approach for multimodal optimization problems. *IEEE Transactions on Evolutionary Computation*, 24(1):114–128.

Wessing, S. (2013). Repair methods for box constraints revisited. In *Applications of Evolutionary Computation - 16th European Conference, EvoApplications 2013*, volume 7835 of *Lecture Notes in Computer Science*, pages 469–478. Springer.

Wu, G., Mallipeddi, R., and Suganthan, P. N. (2017). Problem definitions and evaluation criteria for the CEC 2017 competition on constrained real-parameter optimization. *National University of Defense Technology, Changsha, Hunan, PR China and Kyungpook National University, Daegu, South Korea and Nanyang Technological University, Singapore, Technical Report*.

Yi, W., Chen, Y., Pei, Z., and Lu, J. (2021). Adaptive differential evolution with ensembling operators for continuous optimization problems. *Swarm and Evolutionary Computation*, page 100994.

Zaharie, D. (2009). Influence of crossover on the behavior of differential evolution algorithms. *Applied soft computing*, 9(3):1126–1138.

Zaharie, D. and Micota, F. (2017). Revisiting the analysis of population variance in differential evolution algorithms. In *2017 IEEE Congress on Evolutionary Computation*, pages 1811–1818. IEEE.

Zamuda, A. and Brest, J. (2012). Population reduction differential evolution with multiple mutation strategies in real world industry challenges. In *Swarm and Evolutionary Computation*, pages 154–161. Springer Berlin Heidelberg.

Zhan, Z.-H., Wang, Z.-J., Jin, H., and Zhang, J. (2020). Adaptive distributed differential evolution. *IEEE Transactions on Cybernetics*, 50(11):4633–4647.

Zhang, J. and Sanderson, A. C. (2009). JADE: Adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation*, 13(5):945–958.

Zhao, H., Zhan, Z.-H., Lin, Y., Chen, X., Luo, X.-N., Zhang, J., Kwong, S., and Zhang, J. (2020). Local binary pattern-based adaptive differential evolution for multimodal optimization problems. *IEEE Transactions on Cybernetics*, 50(7):3343–3357.

Zhong, X. and Cheng, P. (2021). An elite-guided hierarchical differential evolution algorithm. *Applied Intelligence*, 51:4962–4983.

Zhou, X.-G., Peng, C.-X., Liu, J., Zhang, Y., and Zhang, G.-J. (2020). Underestimation-assisted global-local cooperative differential evolution and the application to protein structure prediction. *IEEE Transactions on Evolutionary Computation*, 24(3):536–550.

Zuo, M. and Guo, C. (2022). De/current-to-better/1: A new mutation operator to keep population diversity. *Intelligent Systems with Applications*, 14:200063.

A Proofs for statements in Sections 4 and 5

All proofs presented in this Appendix are for `DE/rand/1` mutation with a scale factor $F \in (0, 1]$ in the case of flat objective functions (no selection pressure). The notations used are: x_i - i-th component of an individual from the current population; z_i - i-th component of the DE mutant (which could be in or out of the bounds); $c(z_i)$ - i-th component of the DE mutant corrected by applying an SDIS (where appropriate, the SDIS type is specified as an index, e.g. $c_{\mathtt{sat}}$); $d = z - x$ denotes the DE search direction and $d_c = c(z) - x$ denotes the corrected search direction (where appropriate, the specific SDIS is specified as an index, e.g. $d_{\mathtt{sat}}$); p_v - the probability that a component violates the bounds.

A.1 Bound violation probability for `sat` – Section 4.1

Proposition 1. If $p_v(g)$ denotes the probability of bound violation corresponding to generation g, then, under the assumption that the distribution of the population of elements that are within the bounds (i.e., in (a_i, b_i)) is close to the uniform distribution and `sat` is applied to infeasible individuals, the probability of violation satisfies

$$p_v(g+1) \simeq p_v(g)/2 + (1 - p_v(g))(p_v^2(g)F/2 + (1 - p_v^2(g))F/3) \tag{9}$$

Proof. The analysis is conducted at component level, thus the component index is skipped. Let us consider that the population corresponding to generation g, $P(g)$, consists of three subpopulations: $P(g) = P_w(g) \cup P_{lb}(g) \cup P_{ub}(g)$, corresponding to mutants generated inside the bounding box ($P_w(g)$), placed on the lower bound ($P_{lb}(g)$) and placed on the upper bound ($P_{ub}(g)$). If N denotes the number of elements in $P(g)$, the expected size of $P_w(g)$ is $N(1 - p_v(g))$ and the expected size of $P_{lb}(g)$ and $P_{ub}(g)$ is

$Np_v(g)/2$, respectively (based on the assumption that in the absence of selection pressure, there is no incentive to preferentially violate one of the bounds). To estimate the probability of generating infeasible mutants in generation $g+1$ we consider two cases:

(a) The base individual, x_{r_1}, is on one of the bounds ($x_{r_1} \in P_{lb}(g) \cup P_{ub}(g)$). In this case, the probability of generating an infeasible mutant, $x_{r_1} + F \cdot (x_{r_2} - x_{r_3})$, is $1/2$ as there are no incentives to sample more frequently positive or negative differences. On the other hand, the probability of selecting a base element from the bounds is $p_v(g)$ (under the assumption that the probability of generating through DE mutation individuals which are on the bounds is negligible, i.e. they are generated only by applying the `sat` strategy). Thus in this case $p_v(g+1) = p_v(g)/2$.

(b) The base individual x_{r_1} is strictly between the bounds (event of probability $(1 - p_v(g))$). In this case, we can analyse three subcases:

(i) both x_{r_2} and x_{r_3} belong to the same bound (event of probability $p_v^2(g)/2$): in this case the mutant will be just x_{r_1}, thus feasible;

(ii) x_{r_2} and x_{r_3} belong to different bounds (event of probability $p_v^2(g)/2$): in this case, the probability to generate an infeasible mutant is F; this follows from the remark that Prob$(x_{r_1} + F \cdot (b-a) \in [a,b]) =$ Prob$(x_{r_1} \in [a - F \cdot (b-a), b - F \cdot (b-a)] = (1-F)(b-a)/(b-a) = 1-F$, thus the probability of generating a value out of $[a,b]$ is F.

(iii) x_{r_2} and x_{r_3} belong both to P_w or at most one is on the bound (event of probability $(1 - p_v^2(g))$): in this case, under the assumption that the population of elements belonging to P_w has a distribution which is close to the uniform one, the probability of generating an infeasible mutant is close to $F/3$.

By combining the probabilities corresponding to these complementary events one obtains:

$$p_v(g+1) \simeq \frac{p_v(g)}{2} + (1 - p_v(g)) \left(p_v^2(g) \frac{F}{2} + (1 - p_v^2(g)) \frac{F}{3} \right)$$

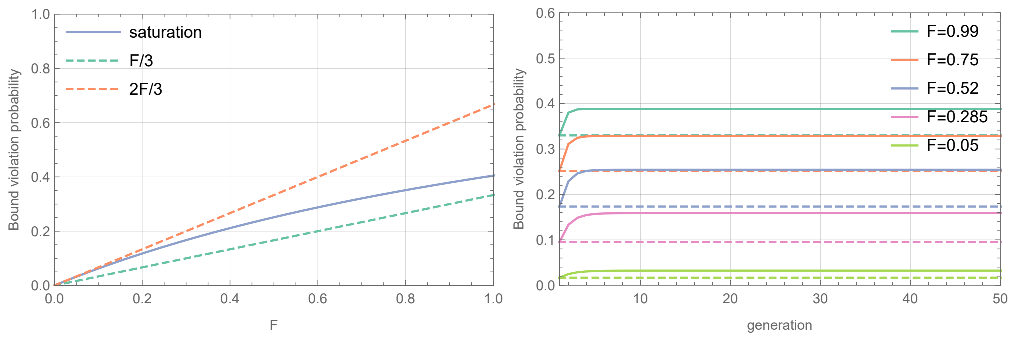


Figure 12: Probability of bounds violation in the case of `sat` (a) limit value (left); (b) evolution during generations (right).

□

Remark. It is easy to see that for $F \in (0, 1]$ the sequence $p_v(g)$ converges to a value between $F/3$ and $2F/3$ (see Fig. 12).

A.2 (Dis)similarity between search directions: cosine similarity for search directions corresponding to **`mir`** and **`tor`** – Section 4.3

Proposition 2. If $0 < F \leqslant 0.5$ and x and $c_{\texttt{mir}}(z)$ belong to the same quadrant, i.e. $(c_{\texttt{mir}}(z_i) - (a_i + b_i)/2)(x_i - (a_i + b_i)/2) \geqslant 0$ for $i = \overline{1, n}$ then $\cos(d, d_{\texttt{mir}}) \geqslant \cos(d, d_{\texttt{tor}})$.

Proof. Since $c_{\texttt{mir}}(z_i) + c_{\texttt{tor}}(z_i) = a_i + b_i$ it follows that $d^T d_{\texttt{mir}} - d^T d_{\texttt{tor}} = \sum_{i=1}^{n}(z_i - x_i)(2c_{\texttt{mir}}(z_i) - (a_i + b_i))$. If $F \leqslant 0.5$ then an infeasible component, z_i, satisfies either $a_i - (b_i - a_i)/2 \leqslant z_i < a_i$ or $b_i < z_i \leqslant b_i + (b_i - a_i)/2$. Thus, when $a_i - (b_i - a_i)/2 \leqslant z_i < a_i \leqslant x_i$ then $c_{\texttt{mir}}(z_i) = 2a_i - z_i \leqslant (a_i + b_i)/2$ meaning that $(z_i - x_i)(2c_{\texttt{mir}}(z_i) - (a_i + b_i)) \geqslant 0$.

In the other case, e.g. $x_i \leqslant b_i < z_i \leqslant b_i + (b_i - a_i)/2$ one have $c_{\texttt{mir}}(z_i) = 2b_i - z_i \geqslant (a_i + b_i)/2$ leading to $(z_i - x_i)(2c_{\texttt{mir}}(z_i) - (a_i + b_i)) \geqslant 0$. Thus, if $F \leqslant 0.5$ the scalar product between the DE search direction and the search direction corresponding to `mir` is larger than that corresponding to `tor` strategy, i.e. $d^T d_{\texttt{mir}} \geqslant d^T d_{\texttt{tor}}$.

On the other hand, when comparing the Euclidean norms of $d_{\texttt{mir}}$ and $d_{\texttt{tor}}$ one obtains:

$$\|d_{\texttt{tor}}\|^2 - \|d_{\texttt{mir}}\|^2 = \sum_{i=1}^{n}[(c_{\texttt{mir}}(z_i) - x_i)^2 - (a_i + b_i - c_{\texttt{mir}}(z_i) - x_i)^2] =$$

$$= \sum_{i=1}^{n} [(a_i + b_i - 2c_{\texttt{mir}}(z_i))(a_i + b_i - 2x_i)] \tag{10}$$

Thus, if $c_{\texttt{mir}}(z_i) - (a_i + b_i)/2$ and $x_i - (a_i + b_i)/2$ have the same sign for all $i = \overline{1, n}$, meaning that $c_{\texttt{mir}}(z)$ and x belong to the same quadrant with respect to $(a_i + b_i)/2$, it follows that $\|d_{\texttt{tor}}\|^2 \geqslant \|d_{\texttt{mir}}\|^2$. By combining this result with the property related to the scalar products it follows that $\cos(d, d_{\texttt{mir}}) \geqslant \cos(d, d_{\texttt{tor}})$ always when $F \leqslant 0.5$ and the corrected and the target elements are in the same quadrant. □

A.3 (Dis)similarity between search directions: cosine similarity analysis for one infeasible component – Section 4.3

Proposition 3. If there is only one infeasible component, e.g. $z_k > b_k$, and the norm of the search direction DE satisfies $\|d\|^2 = \sum_{i=1}^{n}(z_i - x_i)^2 \geqslant 2(z_k - x_k)(z_k - b_k)$, then the cosine similarity between the DE search direction and the direction induced by `sat` is greater than the cosine similarity between the DE search direction and the direction induced by any other SDIS which generate components inside (a_k, b_k).

Proof. If d_c denotes the search direction induced by a SDIS and z_k is the infeasible component, then the cosine between the DE direction, d, and the modified one is:

$$\cos(d, d_c) = \frac{\|d\|^2 + (z_k - x_k)(c(z_k) - x_k) - (z_k - x_k)^2}{\|d\|\sqrt{\|d^2\| + (c(z_k) - x_k)^2 - (z_k - x_k)^2}}$$

$$= \frac{\|d\|^2 + (c(z_k) - z_k)(z_k - x_k)}{\|d\|\sqrt{\|d^2\| + (c(z_k) - x_k)^2 - (z_k - x_k)^2}} \tag{11}$$

To compare $\cos(d, d_c)$ for different SDISs, let us define the function:

$$C(D, \delta_c, \delta) = \frac{(D + \delta(\delta_c - \delta))^2}{D + \delta_c^2 - \delta^2} \tag{12}$$

From Eqs. (11) and (12) it follows that $C(\|d\|^2, c(z_k) - x_k, z_k - x_k)/\|d\|^2 = \cos^2(d, d_c)$. Thus to compare $\cos(d, d_{\texttt{sat}})$ and $\cos(d, d_c)$ it is enough to compare $C(\|d\|^2, c_{\texttt{sat}}(z_k) - x_k, z_k - x_k)$ with $C(\|d\|^2, c(z_k) - x_k, z_k - x_k)$, $c(z_k)$ being a corrected value which belongs to (a_k, b_k). Let us consider the case where $z_k > b_k$, thus $c_{\texttt{sat}}(z_k) = b_k$. To find sufficient conditions that ensure that $\cos(d, d_{\texttt{sat}}) \geqslant \cos(d, d_c)$ one can solve the inequality

$$C(D, \delta_{\texttt{sat}}, \delta) - C(D, \delta_c, \delta) \geqslant 0 \tag{13}$$

with respect to δ_c taking into account the fact that the following conditions are always satisfied:

(i) $D \geqslant \delta^2$, i.e. $\|d\|^2$ is larger or at least equal to the term corresponding to the infeasible component $\delta^2 = (z_k - x_k)^2$;

(ii) $\delta_{\texttt{sat}} > \delta_c$ (since $c(z_k) < b_k$ it follows that $\delta_{\texttt{sat}} = b_k - x_k > c(z_k) - x_k = \delta_c$);

(iii) $\delta_{\texttt{sat}} < \delta$ (if $z_k > b_k$ then $\delta_{\texttt{sat}} = b_k - x_k < z_k - x_k = \delta$);

(iv) $\delta_{\texttt{sat}} \geqslant 0$ (since $x_k \in [a_k, b_k]$ it follows that $\delta_{\texttt{sat}} = b_k - x_k \geqslant 0$).

Using the `Reduce` function from `Wolfram Mathematica 12` to solve the inequality specified in Eq. 13 it follows that the inequality is satisfied at least under the following conditions (depending on the position of the component x_k of the target individual):

(i) if $a_k < x_k \leqslant b_k - (z_k - x_k)$ then $\cos^2(d, d_{\texttt{sat}}) \geqslant \cos^2(d, d_c)$;

(ii) if $b_k - (z_k - x_k) < x_k \leqslant b_k$ and if $\|d\|^2 \geqslant 2\delta(\delta - \delta_{\texttt{sat}}) = 2(z_k - x_k)(z_k - b_k)$ then $\cos^2(d, d_{\texttt{sat}}) \geqslant \cos^2(d, d_c)$;

Since $\cos(d, d_{\texttt{sat}}) \geqslant 0$ it follows that $\cos^2(d, d_{\texttt{sat}}) \geqslant \cos^2(d, d_c)$ implies $\cos(d, d_{\texttt{sat}}) \geqslant \cos(d, d_c)$. A similar result can be obtained when the lower bound is violated, i.e. $z_k < a_k$. □

A.4 Influence of the SDIS on the population diversity – Section 4.4

Proposition 4. If the current population is uniformly distributed on $[0, 1]$, the variance of the mutants corrected by applying `mir` is

$$\text{var}[c_{\texttt{mir}}(Z)] = \frac{F^2}{10} - \frac{F}{4} + \frac{1}{4}.$$

Proof. According to (Ali and Fatti, 2006), the infeasible individuals obtained by applying a `DE/rand/1` (with $F \in [0.5, 1]$) mutation on a uniformly distributed scalar population follow distributions given by:

$$f_{Z_{lb}}(z) = \frac{1}{F}\int_0^{z+F}\left(1 - \frac{x-z}{F}\right)dx = \frac{(F+z)^2}{2F^2} \quad (-F \leqslant z < 0) \tag{14}$$

and

$$f_{Z_{ub}}(z) = \frac{1}{F}\int_{z-F}^{1}\left(1 - \frac{z-x}{F}\right)dx = \frac{(1+F-z)^2}{2F^2} \quad (1 < z \leqslant 1+F) \tag{15}$$

For $F \in (0, 0.5)$, it can be proved that under the same assumption of uniformly distributed individuals in the current population, infeasible mutants follow the same distributions (differences appear only in the distribution of feasible mutants). In order to ensure that Eqs. (14) and (15) correspond to truncated probability distributions on $[-F, 0)$ and $(1, 1+F]$, respectively, each of the functions should be multiplied by $6/F$.

The variance and mean of Z_{lb} and Z_{ub} are var$[Z_{lb}]$ = var$[Z_{ub}]$ = $3F^2/80$ and $\mathbb{E}[Z_{lb}] = -F/4$, $\mathbb{E}[Z_{ub}] = 1 + F/4$, respectively.

Using $Z^{\mathtt{mir}_{lb}} = -Z_{lb}$ and $Z_{ub}^{\mathtt{mir}} = 2 - Z_{ub}$ to denote the random variables corresponding to the individuals corrected by `mir` it follows that $\mathbb{E}[Z_{lb}^{\mathtt{mir}}] = F/4$ and $\mathbb{E}[Z_{ub}^{\mathtt{mir}}] = 1 - F/4$. Since mirroring does not change the variance of the random variable, it follows that var$[Z^{\mathtt{mir}_{lb}}]$ = var$[Z^{\mathtt{mir}_{ub}}] = 3F^2/80$.

The random variable, $c_{\mathtt{mir}}(Z)$, corresponding to the population of corrected individuals can be interpreted as a mixture of variables $Z_{lb}^{\mathtt{mir}}$ and $Z_{ub}^{\mathtt{mir}}$ with mixing weights $w_{lb} = w_{ub} = 1/2$, as in the absence of a selection pressure there should be no difference between the probabilities of violating the lower or the upper bound. The variance of the mixture satisfies Eq. (16).

$$\begin{aligned}\text{var}[c_{\mathtt{mir}}(Z)] \quad = \quad & w_{lb}\text{var}[Z_{lb}^{\mathtt{mir}}] + w_{ub}\text{var}[Z_{ub}^{\mathtt{mir}}] + w_{lb}(\mathbb{E}[Z_{lb}^{\mathtt{mir}}])^2 + w_{ub}(\mathbb{E}[Z_{ub}^{\mathtt{mir}}])^2 - \\ & (w_{lb}\mathbb{E}[Z_{lb}^{\mathtt{mir}}] + w_{ub}\mathbb{E}[Z_{ub}^{\mathtt{mir}}])^2\end{aligned} \tag{16}$$

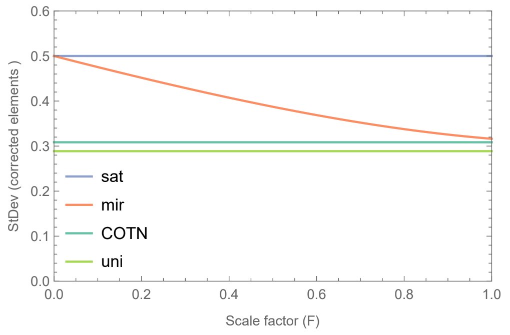


Figure 13: Standard deviation of the population of corrected elements (`sat`, `mir`, `COTN`, `uni`) under the assumption that the distribution of the current population is close to the uniform distribution on $[0, 1]$.

By replacing in Eq. (16) the mixing weights w_{lb} and w_{ub} with $1/2$, the variance and the mean of $Z_{lb}^{\mathtt{mir}}$ and $Z_{ub}^{\mathtt{mir}}$ with the values mentioned above, one obtains that

$$\begin{aligned} \mathrm{var}[c_{\mathtt{mir}}(Z)] &= \frac{1}{2}\left(\frac{3F^2}{40}+\left(\frac{F}{4}\right)^2+\left(1-\frac{F}{4}\right)^2\right)-\frac{1}{4}\left(\frac{F}{4}+1-\frac{F}{4}\right)^2 \\ &= \frac{3F^2}{40}+\frac{F^2}{16}-\frac{F}{4}+\frac{1}{4} \\ &= \frac{F^2}{10}-\frac{F}{4}+\frac{1}{4} \end{aligned} \tag{17}$$

□

Proposition 5. Let us consider that $p \in [0, 1]$ denotes the fraction of infeasible components which violate the lower bound. The variance of the mutants corrected by applying `COTN` is:

$$\mathrm{var}[c_{\mathtt{COTN}}(Z)] = \left(\left(1+\frac{8}{9\pi}-\frac{4}{3}\sqrt{\frac{2}{\pi}}\right)p(1-p)+\frac{\pi-2}{9\pi}\right)(b-a)^2 \tag{18}$$

Proof. Let us consider the distribution probability, $f_{\mathtt{COTN}}^{lb}$, corresponding to corrected components obtained by applying `COTN` in the case when the lower bound has been

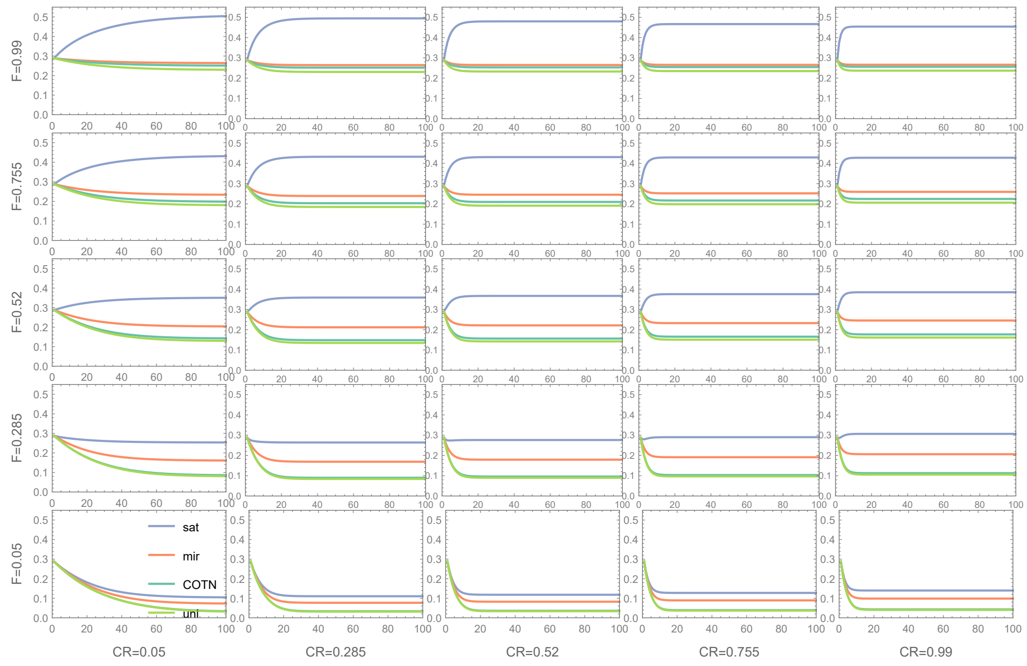


Figure 14: Theoretical influence of SDIS (`sat`, `mir`, `COTN`, `uni`) and of C_r and F on the evolution of the expected standard deviation of the population (`DE/rand/1/bin`, population size of 100, $n = 30$).

violated:

$$f^{lb}_{\texttt{COTN}}(v) = \frac{2}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(v-a)^2}{2\sigma^2}\right), \quad v \in [a, \infty). \tag{19}$$

The mean of this probability distribution is $a + \sigma\sqrt{2/\pi}$ and the variance is $(1 - 2/\pi)\sigma^2$. In the case of the violation of the upper bound, the corrected components have the distribution:

$$f^{ub}_{\texttt{COTN}}(v) = \frac{2}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(b-v)^2}{2\sigma^2}\right), \quad v \in (-\infty, b] \tag{20}$$

for which the mean is $b - \sigma\sqrt{2/\pi}$, and the variance is again $(1 - 2/\pi)\sigma^2$.

The variance of the mixture, computed using the same approach as in Eq. (16), the weight $w_{lb} = p$ for the distribution corresponding to lower bound violation and the weight $w_{ub} = (1 - p)$ for the upper bound violation case, is that given in Eq. (18) which becomes $(1 + 12(1 - \sqrt{2/\pi}))(b - a)^2/36 = 0.095(b - a)^2$ when $p = 1/2$.

□

Remark. The comparison between the standard deviation of the populations of individuals corrected using `sat`, `mir`, `COTN` and `uni` is illustrated in Fig.13. For the same SDISs, Fig. 14 illustrates the evolution of the standard deviation of the entire population over the first 100 generations corresponding to the same values of F and C_r as those used in the experimental analysis presented in section 5.3.

A.5 Analysis of cosine similarity distribution – Section 5.2

Proposition 6. If X and Y are two independent random variables such that their cumulative distribution functions, F_X and F_Y, satisfy $F_X(x) \geqslant F_Y(x)$ then the probability that X is smaller than Y is greater than 0.5, that is, it is more likely that X is smaller than Y than the other way around.

Proof. The cumulative distribution function of $Z = X - Y$ satisfies:

$$F_Z(z) = \int_{-\infty}^{\infty}\int_{-\infty}^{y+z} f_{XY}(x,y)dxdy = \int_{-\infty}^{\infty}\int_{-\infty}^{y+z} f_X(x)f_Y(y)dxdy \tag{21}$$

Since $F_X(x) \geqslant F_Y(x)$ it follows that the probability density functions f_X and f_Y satisfy the same property, that is, $f_X(x) \geqslant f_Y(x)$. On the other hand, the probability that X is smaller than Y is $P(Z \leqslant 0) = F_Z(0)$ which satisfies:

$$\begin{aligned} F_Z(0) &= \int_{-\infty}^{\infty}\int_{-\infty}^{y} f_X(x)f_Y(y)dxdy \\ &\geqslant \int_{-\infty}^{\infty}\int_{-\infty}^{y} f_Y(x)f_Y(y)dxdy \\ &= \int_{-\infty}^{\infty} f_Y(y)\left(\int_{-\infty}^{y} f_Y(x)dx\right)dy \\ &= \int_{-\infty}^{\infty} f_Y(y)F_Y(y)dy = \int_{-\infty}^{\infty} F_Y'(y)F_Y(y)dy \\ &= \frac{1}{2}\int_{-\infty}^{\infty}(F_Y^2(y))'dy = \frac{1}{2}. \end{aligned} \tag{22}$$

Thus $P(X \leqslant Y) \geqslant 0.5$.

□