Artificial Intelligence 314 (2023) 103804

Contents lists available at ScienceDirect

Artificial Intelligence

journal homepage: www.elsevier.com/locate/artint

When move acceptance selection hyper-heuristics outperform Metropolis and elitist evolutionary algorithms and when not [☆]

Andrei Lissovoi [a], Pietro S. Oliveto [a], John Alasdair Warwicker [b]

[a] Department of Computer Science, University of Sheffield, UK
[b] Institute of Operations Research, Karlsruhe Institute of Technology, Germany

A R T I C L E   I N F O

Article history:
Received 30 April 2021
Received in revised form 6 September 2022
Accepted 3 October 2022
Available online 4 October 2022

Keywords:
Hyper-heuristics
Runtime analysis
Non-elitism
Metropolis
Move acceptance operators
Theory

A B S T R A C T

Selection hyper-heuristics (HHs) are automated algorithm selection methodologies that choose between different heuristics during the optimisation process. Recently, selection HHs choosing between a collection of elitist randomised local search heuristics with different neighbourhood sizes have been shown to optimise standard unimodal benchmark functions from evolutionary computation in the optimal expected runtime achievable with the available low-level heuristics. In this paper, we extend our understanding of the performance of HHs to the domain of multimodal optimisation by considering a Move Acceptance HH (MAHH) from the literature that can switch between elitist and non-elitist heuristics during the run. In essence, MAHH is a non-elitist search heuristic that differs from other search heuristics in the source of non-elitism.

We first identify the range of parameters that allow MAHH to hillclimb efficiently and prove that it can optimise the standard hillclimbing benchmark function OneMax in the best expected asymptotic time achievable by unbiased mutation-based randomised search heuristics. Afterwards, we use standard multimodal benchmark functions to highlight function characteristics where MAHH outperforms elitist evolutionary algorithms and the well-known Metropolis non-elitist algorithm by quickly escaping local optima, and ones where it does not. Since MAHH is essentially a non-elitist random local search heuristic, the paper is of independent interest to researchers in the fields of artificial intelligence and randomised search heuristics.

1. Introduction

Selection hyper-heuristics (HHs) are automated algorithm selection methodologies designed to choose which of a set of low-level heuristics to apply in the next steps of the optimisation process [15]. Rather than deciding in advance which heuristic and related parameter settings to apply for a problem, either by manual trial and error in preliminary experiments or using automated algorithm configurators [32,64], the aim behind HHs is to automate the process at runtime. Originally shown to effectively optimise scheduling problems, such as the scheduling of a sales summit and university timetabling [15,16], they have since been successfully applied to a variety of hard combinatorial optimisation problems (see e.g., [7,6,27,40,58,64] for surveys of results).

☆ An extended abstract of this manuscript has appeared at the 2019 Association for the Advancement of Artificial Intelligence Conference (AAAI 2019) [46].

E-mail address: p.oliveto@sheffield.ac.uk (P.S. Oliveto).

https://doi.org/10.1016/j.artint.2022.103804

Selection HHs consist of two separate components: (1) a *heuristic selection method*, often referred to as the *learning mechanism*, to decide which heuristic to be applied in the next step of the optimisation process, and (2) a *move acceptance operator* to decide whether the newly produced search points should be accepted.

The majority of heuristic selection methods in the literature apply machine learning techniques that generate scores for each heuristic based on their past performance. A commonly used method for the purpose is reinforcement learning [15,50,5]. Despite their numerous successful applications, very limited rigorous theoretical understanding of their behaviour and performance is available [43,1].

Recently, it has been proved that a reinforcement learning HH and a simple selection HH called *Random Descent* [15,16], both choosing between elitist randomised local search (RLS) heuristics with different neighbourhood sizes, can respectively optimise the standard unimodal benchmark functions OneMax and LeadingOnes in the best possible runtimes achievable (up to lower order terms), with the available low-level heuristics [22,48]. Since the Random Descent HH does not store information on the past performance of the low-level heuristics, it is necessary to run the selected low-level heuristics for a sufficient amount of time, called the *learning period*, to allow the HH to accurately determine how useful the chosen heuristic is at the current stage of the optimisation process. However, the optimal duration of the learning period may change during the optimisation process. Doerr et al. recently introduced a self-adjusting mechanism and rigorously proved that it allows the HH to track the optimal learning period throughout the optimisation process for LeadingOnes [25]. The optimal asymptotic performance of the same self-adjusting Random Descent HH has recently been shown also for the OneMax and Ridge unimodal benchmark functions [47]. For a survey of the available theoretical results regarding the performance of HHs, see [51].

In this paper we aim to extend the understanding of the behaviour and performance of HHs to multimodal optimisation problems. In order to evaluate their capability at escaping local optima, we consider elitist and non-elitist selection operators (called move acceptance operators) that have been used in the HH literature. Move acceptance operators (also referred to as *selection* operators in the classic evolutionary computation literature) are classified as either *deterministic*, where the same decision is made independent of the stage of the optimisation process, or *non-deterministic*, where different decisions for the same solutions might be made at different stages [56].

Cowling et al. introduced two variants of deterministic move acceptance operators: the elitist OnlyImproving (OI) operator, which only accepts moves that improve the current solution, and the non-elitist AllMoves (AM) operator, which accepts any new solution independent of its quality [15,16]. Another move acceptance operator that has been considered in the literature is the ImprovingandEqual (IE) operator which, in addition to accepting improving solutions, also accepts solutions of equal quality [2,4,55]. In the mentioned works, the acceptance operator remains fixed throughout the run, with the HH only switching between different mutation operators. However, it would be more desirable that HHs are allowed to decide to change the move acceptance operator and hence, the selection pressure at different stages of the optimisation process. For instance, they may use elitist move acceptance in exploitation phases of the search, such as hillclimbing, and non-elitism for exploration, for instance to escape from local optima.

Indeed, Qian et al. analysed a HH that switches between move acceptance operators in the context of multi-objective optimisation [60]. They considered a HH that selects between the elitist (IE) and the strict elitist (OI) move acceptance operators and presented a function where it is necessary to mix the two acceptance operators. Lehre and Özcan presented the only available analysis of a HH which chooses between elitist and non-elitist low level heuristics [43]. In particular, the HH uses the above described OI (strict elitist) and AM (non-elitist) acceptance operators. The considered Move Acceptance HH (MAHH) uses 1-bit flips (i.e., local mutations) as the mutation operator and selects the AM acceptance operator with probability p and the OI acceptance operator with probability $1 - p$. Essentially, the algorithm is a randomised local search (RLS) algorithm that switches between strict elitism, by only accepting improvements, and extreme non-elitism, by accepting any new found solution. For the standard RoyalRoad$_k$ benchmark function from evolutionary computation, which consists of several blocks of $k \geq 2$ bits each that have to be set correctly to observe a fitness improvement, they proved that it is necessary to mix the acceptance operators for MAHH to be efficient, because by only accepting improvements (i.e., $p = 0$) the runtime is infinite due to MAHH not being able to cross the plateaus of equal fitness while by always accepting any move (i.e., $p = 1$), the algorithm simply performs a random search. By choosing the value of the parameter p appropriately they provide an upper bound on the expected runtime of the HH of $O(n^3 \cdot k^{2k-3})$ versus the $O(n \log(n) \cdot (2^k/k))$ expected time required by evolutionary algorithms with standard bit mutation (i.e., each bit is flipped with probability $1/n$) and with the standard selection operator that accepts new solutions if their fitness is at least as good as that of their parent [26]. In particular, just using the IE acceptance operator throughout the run leads to a better performance. Hence, the advantages of switching between selection operators rather than just using one all the time were not evident.

In this paper we present a systematic analysis of the same HH considered by Lehre and Özcan [43] for multimodal optimisation problems,[1] where the considerable advantages of changing the move acceptance operator during the run may be highlighted. In particular, we will increase our understanding of the behaviour and performance of MAHH by providing

[1] In the literature, the term ‘multimodal optimisation’ is also used to refer to optimisation tasks that involve identifying a set of local or global optima (several or all) rather than just the global optimum [59]. In this paper, we use the term to simply refer to the fact that the optimisation functions are not unimodal, i.e., they have local optima in which the algorithms may get trapped. Concerning the term ‘multimodal function’, we use the same definition as in the cited book, i.e., *“Naturally, some objective functions do only possess one optimizer. These are generally considered easier to optimize and are called unimodal. All others are called multimodal”.*

features of multimodal landscapes where it is efficient at escaping local optima and features where it is not. Furthermore, we provide a comparative analysis with the well-studied METROPOLIS non-elitist algorithm [37] and steady-state evolutionary algorithms (EAs), which highlights when MAHH has superior performance, thus should be preferred and when not.

We first perform an analysis of the standard unimodal ONEMAX benchmark function from evolutionary computation to identify the range of parameter values for p that allow MAHH to hillclimb, and hence to locate local optima efficiently. In particular, we prove that for any $p = O((\log\log n)^{1-\varepsilon}/n)$ and any constant $\varepsilon > 0$, MAHH is asymptotically as efficient as the best unary unbiased (i.e., mutation-based) randomised search heuristic [44] even though it does not rely on elitism. On the other hand, we show that for larger $p = \omega(\sqrt{n}\log n)/n)$, MAHH cannot optimise any function with unique optimum in polynomial time with high probability.

Afterwards we highlight the power of MAHH by analysing its performance for standard benchmark function classes chosen because they allow us to isolate important properties of multimodal optimisation landscapes. Firstly, we consider the CLIFF_d class of functions, consisting of local optima that, once escaped, allow the identification of a new slope of increasing fitness. For this class of functions, we rigorously prove that MAHH efficiently escapes the local optima and on the hardest instances of the function class for elitist search heuristics [12], it even achieves the best possible expected runtime of $O(n\log n)$ (for unary - mutation only - unbiased randomised search heuristics) for any problem with up to a polynomial number of optima [3]). Thus, we prove that it considerably outperforms established elitist and non-elitist evolutionary algorithms, including the METROPOLIS algorithm. We then consider the standard JUMP_m multimodal instance class of functions to provide an example of fitness landscape where to identify a new slope with increasing gradient towards the optimum is as hard as possible. While MAHH has a runtime that is exponential in the size of the basin of attraction, it is efficient for instances of moderate jump size (i.e., constant) where it still considerably outperforms METROPOLIS which requires exponential time in the problem size with overwhelming probability independent of the basin's size.

The superior performance of MAHH over METROPOLIS for the considered function classes is mainly due to the large fitness value difference between the local optima and their neighbouring solutions. To this end, we design a smoother function class, called $\text{CLIFFJUMP}_{d,r,s}$, for which the sizes of the basins of attraction of both local and global optima and the steepness of the negative slope that has to be traversed to escape from the local optima can be tuned. Our aim is to identify basin-of-attraction characteristics where MAHH outperforms METROPOLIS and vice-versa. $\text{CLIFFJUMP}_{d,r,s}$ combines aspects of both CLIFF_d and JUMP_m. From the local optima, a 'JUMP_m' slope returning to the local optima has to be overcome to identify a CLIFF_d slope leading to the global optimum. If the negative slope leading away from the local optima decreases in fitness gently, then METROPOLIS can be configured to always outperform any variant of the HH. Conversely, if the slope is steep, then the HH can be configured to be faster than any variant of METROPOLIS. Hence, the analysis shows that MAHH is preferable on rugged landscapes with steep changes in fitness, while METROPOLIS is more efficient on smoother 'well-behaved' landscapes where neighbouring points have similar fitness values. However, while MAHH is always efficient if the length of the negative slope is not too large independent of steepness of the slope, METROPOLIS is inefficient even when the slope is small if the descending gradient is not gentle. As a corollary, the function class allows us to point out what kinds of basin of attraction are overcome more efficiently by the non-elitist hyper-heuristic compared to elitist evolutionary algorithms using standard bit mutation. In particular, for small basins of attraction located at a super-constant distance from the optimum, non-elitism may make a difference between small polynomial and super-polynomial runtimes, including exponential ones.

We use this insight to complete the picture by presenting fitness landscape characteristics where METROPOLIS is efficient while MAHH is not. The benchmark function we consider has a very gentle slope with a smooth fitness-decreasing gradient surrounded by points of considerably lower fitness. While METROPOLIS is efficient for the function because it can follow the gentle path by rejecting the points of lower fitness, the function is deceptive for MAHH since it is likely to accept worsening solutions that are not on the path resulting in exponential optimisation time with overwhelming probability. Since MAHH is essentially a non-elitist random local search heuristic that differs from other search heuristics in the source of non-elitism, the paper is of general interest outside the parameter control [21] and hyper-heuristics community, in particular to researchers in randomised search heuristics and artificial intelligence in general (see, e.g., [36,10,18] for other analyses emphasising the effectiveness of non-elitism).

The rest of the paper is structured as follows. In the next section we formally introduce the move-acceptance hyper-heuristics (MAHH_{OI} and MAHH_{IE}), the problem classes and the mathematical analysis tools used in the rest of the paper. In Section 3, we analyse MAHH_{OI} on the unimodal benchmark function ONEMAX. In Sections 4 and 5, we analyse MAHH_{OI} on the multimodal CLIFF_d and JUMP_m functions, comparing its expected runtime with that of the well known METROPOLIS algorithm. Section 6 analyses the ability of MAHH_{OI} and METROPOLIS to escape basins of attraction with tuneable gradients and size. In Section 7, we analyse an example function where METROPOLIS provably outperforms MAHH_{OI}. We finish the paper with some conclusions on when MAHH performs well and when it doesn't, as well as offering some promising avenues for future research.

Compared to its conference version [46], this manuscript has been considerably extended. We include all proofs that were omitted due to space constraints, and many theorems have been strengthened. Furthermore, sections 6 and 7 are completely new additions.

Algorithm 1 Move Acceptance Hyper-Heuristic (MAHH_{OI}) [43].

```
1: Choose $x \in \{0,1\}^n$ uniformly at random
2: while termination criteria not satisfied do
3:     $x' \leftarrow$ FlipRandomBit($x$)
4:     Choose $r \in [0,1]$ uniformly at random
5:     if $r \le p$ then $x \leftarrow x'$                // AM
6:     else $\Delta f \leftarrow f(x') - f(x)$
7:         if $\Delta f > 0$ then $x \leftarrow x'$         // OI
```

Algorithm 2 Metropolis algorithm [49].

```
1: Choose $x \in \{0,1\}^n$ uniformly at random, set $\alpha(n) \ge 1$
2: while termination criteria not satisfied do
3:     $x' \leftarrow$ FlipRandomBit($x$)
4:     $\Delta f \leftarrow f(x') - f(x)$
5:     if $\Delta f \ge 0$ then $x \leftarrow x'$
6:     else choose $r \in [0,1]$ uniformly at random
7:         if $r \le \alpha(n)^{\Delta f}$ then $x \leftarrow x'$
```

Algorithm 3 $(1+1)$ Evolutionary algorithm.

```
1: Choose $x \in \{0,1\}^n$ uniformly at random
2: while termination criteria not satisfied do
3:     $x' \leftarrow$ flip each bit of $x$ independently with probability $1/n$
4:     $\Delta f \leftarrow f(x') - f(x)$
5:     if $\Delta f \ge 0$ then $x \leftarrow x'$
```

2. Preliminaries

In this section, we will formally introduce the hyper-heuristic algorithms and the problem classes we will analyse in this paper, and briefly state some widely-known mathematical tools for the runtime analysis of randomised search heuristics that we will use throughout the paper.

2.1. Algorithms

We will analyse the Move Acceptance hyper-heuristic previously considered by Lehre and Özcan [43]. In each iteration, one bit chosen uniformly at random will flip and with probability p, the AllMoves (AM) acceptance operator is used, while with probability $1-p$, the OnlyImproving (OI) acceptance operator is used. Algorithm 1 shows its pseudocode.

We also consider the MAHH_{IE} variant of Algorithm 1 whereby the ImprovingAndEqual (IE) acceptance operator (which accepts all moves which either increase or maintain the current fitness) is used instead of the OI operator. Thus, in MAHH_{IE}, the AM acceptance operator is used with probability p, and the IE acceptance operator is used with probability $1-p$. However, for the problem classes we consider, changing the number of 1-bits in the bit-string by 1 (as is done by the FlipRandomBit mutation operator of Algorithm 1) will always change the fitness value, i.e., there are no plateaus of constant fitness. Therefore, we point out that, given these conditions, all the statements made in the paper for the MAHH_{OI} hyper-heuristic will also hold for the MAHH_{IE} hyper-heuristic. Thus, in the rest of the paper, we will focus the analysis on the MAHH_{OI} hyper-heuristic.

Throughout this paper, we will compare the performance of the MAHH_{OI} hyper-heuristic with the well-known Metropolis algorithm [49] which also allows for global exploration through non-elitism, and with a standard elitist evolutionary algorithm ($(1+1)$ EA) which uses global mutations to escape from local optima [39]. While all local mutations which find improving solutions are accepted, Metropolis also accepts worsening moves with some non-zero probability. Suppose the parent solution x has fitness $f(x)$, and the offspring solution y has a fitness $f(y)$, where $f(y) - f(x) = \Delta f < 0$. Metropolis accepts the worsening move with probability $\alpha(n)^{\Delta f}$, for some function $\alpha(n) \ge 1$. The pseudocode for Metropolis is shown in Algorithm 2. The $(1+1)$ EA only accepts solutions of non-decreasing fitness (i.e., elitism) and flips each bit with a fixed probability (i.e., $1/n$) so that it can reach any area of the search space in a single step, while the probability of reaching a new point decreases with the increased Hamming distance to the current point. The pseudocode for the $(1+1)$ EA is shown in Algorithm 3.

2.2. Benchmark function classes

We will start by considering standard benchmark function classes defined over bit strings of length n commonly used in the theory of randomised search heuristics to evaluate their performance. These problem classes are artificially constructed with the purpose of reflecting and isolating common difficulty profiles that are known to appear in classical combinatorial optimisation problems and are expected to appear in real-world optimisation.

The OneMax problem class is a class of unimodal functions which provide a consistent fitness gradient leading to the global optimum. The class displays the typical function optimisation feature that improving solutions are harder to identify

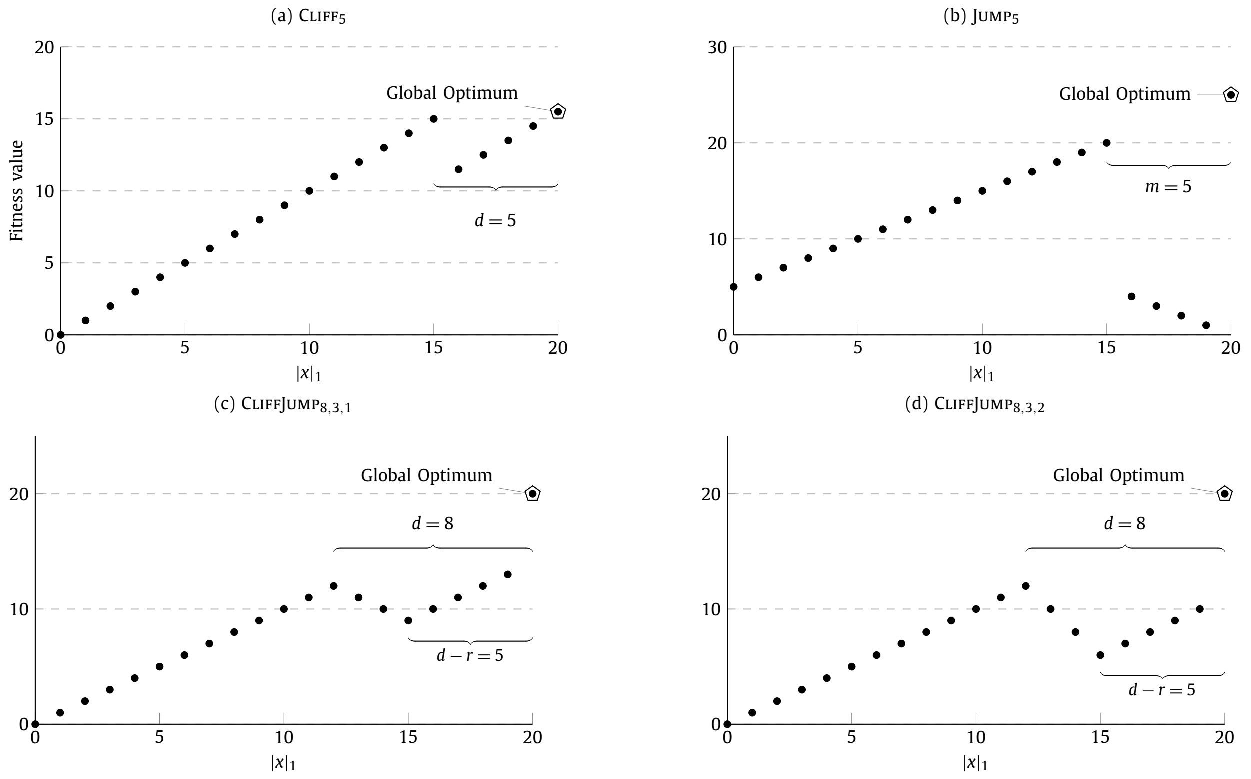

Fig. 1. Examples of the Cliff_d, Jump_m, and $\text{CliffJump}_{d,r,s}$ benchmark functions for $n = 20$.

as the optimum is approached. It is generally used to evaluate and validate the hillclimbing performance of randomised search heuristics. The function is defined as follows:

$$\text{OneMax}(x) := \sum_{i=1}^{n} x_i.$$

In the above definition, the global optimum is placed in the 1^n bit-string for convenience of the analysis, i.e., the fitness increases with the number of 1-bits. The results we derive will hold for any instance in the function class, i.e., the optimum may be any bit string and the fitness function returns the number of matching bits. This class of functions is known to be the easiest among all functions with a unique global optimum for unary unbiased black box algorithms (i.e., mutation-based EAs) [44].

We also consider the Cliff_d and Jump_m multimodal problem classes, where the optimisation algorithm will need to escape from local optima in order to reach the global optimum. The two classes differ in whether the fitness function guides the search towards or away from the global optimum once the algorithm has escaped the local optimum.

The Cliff_d class of functions was originally proposed as an example where non-elitist evolutionary algorithms outperform elitist ones [36]. Functions within the class generally lead the optimisation process to a local optimum, from which a fitness-decreasing mutation can be taken to find another fitness-improving slope leading to the global optimum. An example instance is shown in Fig. 1a. We define the Cliff_d class of functions (for $1 < d < n/2$) as follows:

$$\text{Cliff}_d(x) := \begin{cases} \text{OneMax}(x) & \text{if } |x|_1 \le n - d, \\ \text{OneMax}(x) - d + 1/2 & \text{otherwise.} \end{cases}$$

As for OneMax, the global optimum is placed at the 1^n bit string to simplify notation. The parameter d controls both the fitness decrease from the local optimum to the lowest point on the slope leading to the global optimum and the length of this second slope. This class of problems captures real-world problems where local optima have narrow basins of attraction: if it is able to escape from the local optimum, a search heuristic has good chances of identifying a new basin of attraction.

The Jump_m class of functions is similar to Cliff_d, but has both fitness gradients leading towards the local optima: thus, to find the global optimum, search algorithms should not only take a fitness-decreasing mutation to escape the local optimum, but also disregard the fitness gradient in further iterations. An example instance is shown in Fig. 1b. We define the Jump_m class of functions (for $1 < m < n/2$) as follows:

$$\textsc{Jump}_m(x) := \begin{cases} n+m & \text{if } |x|_1 = n, \\ m + \textsc{OneMax}(x) & \text{if } |x|_1 \le n-m, \\ n - \textsc{OneMax}(x) & \text{otherwise.} \end{cases}$$

As for OneMax, the global optimum is placed at the 1^n bit string to simplify notation. Compared to $\textsc{Cliff}_d$, this class of functions features a local optimum with a much broader basin of attraction, making it more difficult for the search heuristics to escape the basin and locate the global optimum.

Additionally, we define a function class which combines the features of both $\textsc{Cliff}_d$ and $\textsc{Jump}_m$, allowing both the local and global optima to have basins of attraction in the region of the search space between the two optima. Thus, to optimise this function, which we name $\textsc{CliffJump}_{d,r,s}$, a search algorithm employing local mutations may need to take multiple steps away from the local optimum before a slope leading towards the global optimum can be found. Example instances are shown in Figs. 1c and 1d for $s=1$ and $s=2$. We define this class of functions (for $1 \le r < d < n/2$ and $s > 0$) as follows:

$$\textsc{CliffJump}_{d,r,s}(x) := \begin{cases} n & \text{if } |x|_1 = n, \\ \textsc{OneMax}(x) & \text{if } |x|_1 \le n-d, \\ \textsc{OneMax}(x) - r - rs & \text{if } |x|_1 \ge n-d+r, \\ (n-d) - s \cdot (\textsc{OneMax}(x) + d - n) & \text{otherwise.} \end{cases}$$

The global optimum is placed at the 1^n string, while the d and r parameters control the lengths of the positive and negative slopes following the local optimum: after reaching the local optimum, r mutations, each decreasing the fitness by s, must be taken by a local search heuristic before a fitness-increasing slope of length $(d-r)$ becomes accessible.

We further consider the following function (which we rename GentleNegativeSlope), which was originally introduced by Jansen and Wegener as an example of fitness landscape where Metropolis considerably outperforms the $(1+1)$ EA [38].

$$\textsc{GentleNegativeSlope}(x) := \begin{cases} 2^n \cdot n + 1 & \text{if } x = 0^n, \\ 2^n \cdot n - (n-i) & \text{if } x = 1^i 0^{n-i}, i \in \{1, \dots, n\}, \\ 2^n \cdot \textsc{OneMax}(x) & \text{otherwise.} \end{cases}$$

GentleNegativeSlope (GNS) is a deceptive function that will lead most algorithms away from the global optimum at 0^n. There is a steep OneMax style path to the 1^n bit-string followed by a ridge with gently decreasing fitness towards the global optimum.

2.3. Mathematical analysis tools

We now present some well known drift analysis theorems often used to bound the expected runtime of randomised search heuristics. Drift analysis is a very general mathematical technique that allows to make statements regarding the long term performance (i.e., the runtime) of a randomised search heuristic by analysing the expected progress that the algorithm makes in one step. All that is required to apply drift analysis is a potential function (sometimes called a ‘distance function’) to estimate the distance of the current solution to a target area of the search space (e.g., the global optimum), and a bound on the expected one-step change (referred to as the ‘drift’) that the algorithm makes with respect to the potential function conditional on the current distance from the target. Hence, given that at time step t the distance from the target is X_t, analysing the drift requires providing bounds on the quantity $E(X_t - X_{t+1} \mid X_t)$ to quantify the expected one-step progress of the algorithm. For a gentle introduction to drift analysis we refer to [42], while for a more extensive overview we refer the reader to Lengler’s recent book chapter [45].

We will apply the following theorems throughout the paper for our analyses using the Hamming distance as a measure of distance to the optimum (or another set of target solutions), and write $\Delta(i)$ to refer to the drift conditioned on the parent solution containing i 1-bits. The following Additive Drift Theorem provides upper and lower bounds on the expected runtime given, respectively, lower and upper bounds on the drift which hold throughout the process.

Theorem 1 *(Additive Drift Theorem [34]). Let $\{X_t\}_{t\ge 0}$ be a sequence of random variables over a finite set of states $S \subseteq \mathbb{R}_0^+$ and let T be the random variable that denotes the first point in time for which $X_t = 0$. If there exist $\delta_u \ge \delta_l > 0$ such that for all $t \ge 0$, we have*

$$\delta_u \ge E(X_t - X_{t+1} \mid X_t) \ge \delta_l,$$

then the expected optimisation time $E(T)$ satisfies

$$\frac{X_0}{\delta_u} \le E(T \mid X_0) \le \frac{X_0}{\delta_l}, \quad \textit{and}$$
$$\frac{E(X_0)}{\delta_u} \le E(T) \le \frac{E(X_0)}{\delta_l}.$$

When the expected drift is negative in a non-trivial region of the search space, exponential lower bounds on the runtime can be derived using the Negative Drift Theorem.

Theorem 2 *(Negative Drift Theorem [52,53]). Let X_t, $t \geq 0$, be the random variables describing a Markov process over some state space, and let $\delta_t(i) := (X_t - X_{t+1} \mid X_t = i)$ for $i \in S$ and $t \geq 0$. Suppose there exist an interval $[a, b]$ of the state space and two constants $\delta, \epsilon > 0$ such that for all $t \geq 0$ the following two conditions hold:*

1. $E(\delta_t(i)) \leq -\epsilon$ *for* $a < i < b$,
2. $\Pr(|\delta_t(i)| \geq j) \leq 1/(1+\delta)^j$ *for* $i > a$ *and* $j \geq 0$.

Then there is a constant $c^* > 0$ such that for $T^* := \min\{t \geq 0 : X_t \leq a \mid X_0 \geq b\}$ it holds $\Pr(T^* \leq 2^{c^*(b-a)}) = 2^{-\Omega(b-a)}$.

We will also use the Negative Drift Theorem with Scaling which allows the negative drift ε to be sub-constant in magnitude.

Theorem 3 *(Negative Drift Theorem with Scaling [54]). Let X_t, $t \geq 0$, be real-valued random variables describing a stochastic process over some state space. Suppose that there exist an interval $[a, b] \subseteq \mathbb{R}$ and, possibly depending on $\ell := b - a$, a drift bound $\varepsilon := \varepsilon(\ell) > 0$, as well as a scaling factor $r := r(\ell)$ such that for all $t \geq 0$ the following conditions hold:*

1. $E(X_{t+1} - X_t \mid X_0, \ldots, X_t; a < X_t < b) \geq \varepsilon$,
2. $\Pr(|X_{t+1} - X_t| \geq jr \mid X_0, \ldots, X_t; a < X_t) \leq e^{-j}$ *for all* $j \in \mathbb{N}_0$,
3. $1 \leq r^2 \leq \varepsilon\ell/(132\log(r/\varepsilon))$.

Then for the first hitting time $T^* := \min\{t \geq 0 : X_t \leq a \mid X_0 > b\}$ it holds that $\Pr(T^* \leq e^{\varepsilon\ell/(132r^2)}) = O(e^{-\varepsilon\ell/(132r^2)})$.

3. Unimodal optimisation: hillclimbing

We begin the study of MAHH$_{\text{OI}}$ by analysing its performance on unimodal functions. Since accepting worsening moves is never required in this setting, the hyper-heuristic cannot outperform elitist algorithms. Nevertheless, Theorem 7 shows that MAHH$_{\text{OI}}$ can still be very efficient when optimising OneMax, even with $p > 0$. We first introduce Theorem 4 and Theorem 5, which show that the parameter p should not be too large. Theorem 6 subsequently states that the expected runtime of MAHH$_{\text{OI}}$ for OneMax is a lower bound for its expected runtime on all functions with a unique global optimum.

For OneMax, the larger the value of p, the greater the probability of accepting moves away from the global optimum during the optimisation process, and the greater the expected runtime of the hyper-heuristic. For constant values of p, the standard Negative Drift Theorem (Theorem 2) can be applied directly to show that an exponential number of iterations is required with exponentially high probability.

Theorem 4. *The runtime of MAHH$_{OI}$ on* OneMax, *with $p = \Theta(1)$, is at least $2^{\Omega(n)}$ with probability at least $1 - 2^{-\Omega(n)}$.*

Proof. Let i denote the number of 1-bits in the current bit-string, and consider the expected change in the Hamming distance (i.e., drift) to the optimum (which for OneMax is equivalent to the expected change in solution fitness) in an iteration of MAHH$_{\text{OI}}$:

$$\Delta(i) = \frac{n-i}{n} - p \cdot \frac{i}{n} = -\frac{i + pi - n}{n}, \tag{1}$$

as improving mutations (which require flipping one of the $n-i$ remaining 0-bits) are accepted by both acceptance operators, while worsening mutations (flipping one of the i 1-bits) are accepted only by the AM operator, chosen with probability p.

Let $p > 0$ be a constant. We have that for $i \geq \frac{1+C}{1+p} \cdot n$, $\Delta(i) \leq -C$, for some constant C, with $p > C > 0$. Hence, for a region of size $(n-1) - \frac{1+C}{1+p} \cdot n = \Omega(n)$, the drift is negative (at most $-C$). Since MAHH$_{\text{OI}}$ is a local search algorithm, this region of negative drift cannot be escaped by a large jump. By the Negative Drift Theorem (Theorem 2), the runtime in this case will be at least $2^{\Omega(n)}$ with overwhelming probability $1 - 2^{-\Omega(n)}$. □

For smaller values of p, yet still too large, we apply the negative drift theorem with scaling (Theorem 3).

Theorem 5. *The runtime of MAHH$_{OI}$ for* OneMax, *with $p = \omega((\sqrt{n}\log n)/n)$, is at least $n^{\omega(1)}$ with probability at least $1 - n^{-\omega(1)}$.*

Proof. To prove this result, we will apply the Simplified Drift Theorem with Scaling (Theorem 3), which handles regions of sub-constant negative drift.

Recall Equation (1) from the proof of Theorem 4: the drift in one iteration for MAHH$_{\text{OI}}$ for OneMax, when the current solution contains i 1-bits, is

$$\Delta(i) = \frac{n-i}{n} - p \cdot \frac{i}{n} = -\frac{i+pi-n}{n}.$$

We will consider the drift in the region of $n - c\sqrt{n} \le i < n$ of length $\ell = c\sqrt{n}$, where $c > 0$ is a constant. Let $p = c^+ \cdot (\sqrt{n}\log n)/n$, where $c^+ = \omega(1)$. As $\Delta(i)$ decreases with i, the negative drift is weakest at $i = n - \ell$:

$$\begin{aligned}\Delta(n-\ell) &= -\frac{(n-\ell)+(n-\ell)p-n}{n} = -\frac{np-\ell p-\ell}{n}\\ &= -\left(p - \frac{c\sqrt{n}p}{n} - \frac{c\sqrt{n}}{n}\right) = -\left(p - o(p) - O(n^{-1/2})\right) = -\Omega(p)\end{aligned}$$

and thus $\varepsilon = \omega\left(\frac{\sqrt{n}\log n}{n}\right)$ satisfies the drift condition of the Simplified Drift Theorem with Scaling (Theorem 3).

The second condition, forbidding large jumps, is satisfied by choosing $r = 2$ and verifying that for all $j \in \mathbb{N}_0$, the probability that the magnitude of the fitness change is at least jr is at most e^{-j}, which is trivially true: for $j = 0$, $e^{-j} = 1$, and for $j \ge 1$, the probability that the distance to the optimum changes by at least $jr \ge 2$ is 0, as the mutation operator used by MAHH$_{\text{OI}}$ only flips one bit per iteration.

Finally, we need to verify the following condition, $1 \le r^2 \le \varepsilon\ell/(132\log(r/\varepsilon))$. To this end, we note that

$$\varepsilon\ell = \omega\left(\frac{\sqrt{n}\log n}{n}\right) \cdot c\sqrt{n} = \omega(\log n),$$

and

$$132\log(r/\varepsilon) = 132\log\left(\frac{2\sqrt{n}}{\omega(\log n)}\right) \le 132\log(2\sqrt{n}) \le 66\log(n) + O(1).$$

Thus $\varepsilon\ell/(132\log(r/\varepsilon)) = \omega(1)$ is greater than $r^2 = 4$ for sufficiently large n.

Having verified that all of the conditions of the Simplified Drift Theorem with Scaling are satisfied, we apply the theorem, concluding that the probability that the optimum is found within $e^{\varepsilon\ell/(132r^2)} = n^{\omega(1)}$ iterations is at most $O(e^{\varepsilon\ell/(132r^2)}) = n^{-\omega(1)}$. □

We remark that the $\omega((\sqrt{n}\log n)/n)$ lower bound on the probability of applying the AM selection operator is the lowest such bound for which the Negative Drift Theorem with scaling yields a super-polynomially high probability of requiring a super-polynomial number of iterations to optimise OneMax. We note that $\varepsilon\ell = \omega(\log n)$ is required for Theorem 3 to yield a super-polynomial runtime bound, and further reducing p would require reductions to both the drift bound ε and the length of the negative drift region ℓ. Closing the gap to the upper bound on this probability for which a positive result is presented in Theorem 7 would thus require other proof techniques to be applied.

The following theorem proves that MAHH$_{\text{OI}}$ in the same setting cannot solve any function with a unique global optimum in polynomial time. Thus we provide a pretty general lower bound on the value of parameter p for efficient optimisation. We follow the proof idea of [23, Theorem 9], which showed that the expected runtime of the (1 + 1) EA on OneMax is a lower bound on its expected runtime for any function with a unique global optimum. [65, Theorem 10] further proved the same result for arbitrary mutation-based EAs with mutation probability $1/n$.

Theorem 6. *The expected runtime of MAHH$_{\text{OI}}$ for any function with a unique global optimum is at least as large as the expected runtime of MAHH$_{\text{OI}}$ for* OneMax *with the same setting for p.*

Proof. Let f denote any function with a unique global optimum and without loss of generality we set the unique global optimum of f at the bit-string 1^n. We will prove that the expected runtime of MAHH$_{\text{OI}}$ for f is at least as large as the runtime of MAHH$_{\text{OI}}$ for OneMax. Formally, we aim to show that

$$E^{f}_{\text{MAHH}_{\text{OI}}} \ge E^{\text{OneMax}}_{\text{MAHH}_{\text{OI}}}. \tag{2}$$

Let $E^{f}_{\text{MAHH}_{\text{OI}}}(i)$ denote the minimum expected time required by MAHH$_{\text{OI}}$ to find the global optimum of f, given that MAHH$_{\text{OI}}$ has only sampled solutions x with $|x|_1 \le i$ so far. MAHH$_{\text{OI}}$ only samples Hamming neighbours and hence must sample at least one solution with $|x|_1 = j$ for all $j > i$ before reaching the global optimum. By definition,

$$E^{f}_{\text{MAHH}_{\text{OI}}}(n) \le E^{f}_{\text{MAHH}_{\text{OI}}}(n-1) \le \cdots \le E^{f}_{\text{MAHH}_{\text{OI}}}(0).$$

We further define $\widetilde{E}^{\text{OneMax}}_{\text{MAHH}_{\text{OI}}}(i)$ be defined the same as $E^{\text{OneMax}}_{\text{MAHH}_{\text{OI}}}(i)$ yet with the added constraint that a solution with $|x|_1 = i$ has been sampled. By definition, $\widetilde{E}^{\text{OneMax}}_{\text{MAHH}_{\text{OI}}}(i) \ge E^{\text{OneMax}}_{\text{MAHH}_{\text{OI}}}(i)$.

We will verify the claim in Equation (2) by proving that $E^f_{\mathrm{MAHH_{OI}}}(i) \geq \widetilde{E}^{\textsc{OneMax}}_{\mathrm{MAHH_{OI}}}(i)$ holds for all i. We use proof by induction. Firstly, note that $E^f_{\mathrm{MAHH_{OI}}}(n) = \widetilde{E}^{\textsc{OneMax}}_{\mathrm{MAHH_{OI}}}(n) = 0$, and we assume that $E^f_{\mathrm{MAHH_{OI}}}(j) \geq \widetilde{E}^{\textsc{OneMax}}_{\mathrm{MAHH_{OI}}}(j)$ for all $j > i$.

Since $\mathrm{MAHH_{OI}}$ uses local search with a neighbourhood size of one, the number of 1-bits in the solution can either decrease by one (if a worsening move is accepted by the AM operator), stay the same (if a worsening move is rejected by the OI operator) or increase by one. If the number of 1-bits increases, the expected optimisation time is at least $E^f_{\mathrm{MAHH_{OI}}}(i+1)$. Let Y denote the number of 1-bits in the solution after the mutation is accepted or rejected; using the induction hypothesis,

$$\begin{aligned} E^f_{\mathrm{MAHH_{OI}}}(i) &\geq 1 + \Pr(Y = i+1) \cdot E^f_{\mathrm{MAHH_{OI}}}(i+1) + \Pr(Y \leq i) \cdot E^f_{\mathrm{MAHH_{OI}}}(i) \\ &\geq 1 + \Pr(Y = i+1) \cdot \widetilde{E}^{\textsc{OneMax}}_{\mathrm{MAHH_{OI}}}(i+1) + \Pr(Y \leq i) \cdot E^f_{\mathrm{MAHH_{OI}}}(i). \end{aligned}$$

Hence,

$$E^f_{\mathrm{MAHH_{OI}}}(i) \geq \frac{1 + \Pr(Y = i+1) \cdot \widetilde{E}^{\textsc{OneMax}}_{\mathrm{MAHH_{OI}}}(i+1)}{1 - \Pr(Y \leq i)}.$$

Furthermore, for $\mathrm{MAHH_{OI}}$ on OneMax we have,

$$\widetilde{E}^{\textsc{OneMax}}_{\mathrm{MAHH_{OI}}}(i) = 1 + \Pr(Y = i+1) \cdot \widetilde{E}^{\textsc{OneMax}}_{\mathrm{MAHH_{OI}}}(i+1) + \Pr(Y \leq i) \cdot \widetilde{E}^{\textsc{OneMax}}_{\mathrm{MAHH_{OI}}}(i),$$

and hence,

$$\widetilde{E}^{\textsc{OneMax}}_{\mathrm{MAHH_{OI}}}(i) = \frac{1 + \Pr(Y = i+1) \cdot \widetilde{E}^{\textsc{OneMax}}_{\mathrm{MAHH_{OI}}}(i+1)}{1 - \Pr(Y \leq i)}.$$

Therefore, we have proven that $E^f_{\mathrm{MAHH_{OI}}}(i) \geq \widetilde{E}^{\textsc{OneMax}}_{\mathrm{MAHH_{OI}}}(i) \geq E^{\textsc{OneMax}}_{\mathrm{MAHH_{OI}}}(i)$. Since $\mathrm{MAHH_{OI}}$ is initialised uniformly at random regardless of the fitness function used, the distribution of the number of 1-bits in the initial solution is the same. Hence, the theorem holds. □

Theorem 6 combined with the statements of Theorem 4 and Theorem 5 shows that the expected runtime of $\mathrm{MAHH_{OI}}$ on any function with unique global optimum is at least $n^{\Omega(\log n)}$ if $p \geq (\sqrt{n}\log^2 n)/n$ and at least $2^{\Omega(n)}$ if $p = \Theta(1)$.

We now provide an upper bound on p for which the hyper-heuristic is efficient at hillclimbing OneMax.

Theorem 7. *The expected runtime of $\mathrm{MAHH_{OI}}$ for* OneMax*, with $p = O((\log\log n)^{1-\epsilon}/n)$, for any constant $\epsilon > 0$, is $O(n \log n)$. With $p = O((\log n)^{1-\epsilon}/n)$, for any constant $\epsilon > 0$, the expected runtime of $\mathrm{MAHH_{OI}}$ for* OneMax *is $o(n^2 \log n)$.*

Proof. Let Y_t denote the current solution at iteration t, $h(Y_t)$ be its Hamming distance to the OneMax optimum, $g : \mathbb{N}_0 \to \mathbb{R}_{\geq 0}$ be an injective function with $g(0) = 0$, and g^{-1} the inverse function of g. To prove this result, we will apply the additive drift theorem on the process $X_t := g(h(Y_t))$, bounding the expected number of steps before $X_t = 0$ for the first time (and hence $\mathrm{MAHH_{OI}}$ has constructed an optimal solution for the first time).

Consider the drift $\Delta(i) := E(X_t - X_{t+1} \mid g^{-1}(X_t) = i)$, i.e. the expected decrease of the X_{t+1} value when the current solution is a Hamming distance of i away from the optimum. If $\mathrm{MAHH_{OI}}$ mutation produces an improvement, it will be accepted regardless of which selection operator is used and hence decrease $h(Y_t)$ by 1. If $\mathrm{MAHH_{OI}}$ mutation produces a worsening, it will only be accepted if the AM operator is applied (with probability p), in which case $h(Y_t)$ will increase by 1; otherwise, $h(Y_t)$ and hence also X_t will remain unchanged. Combining the above yields:

$$\Delta(i) = \frac{i}{n} \cdot (g(i) - g(i-1)) - \frac{n-i}{n} \cdot p \cdot (g(i+1) - g(i)). \tag{3}$$

Let $g(x) = \sum_{i=1}^{x} (\frac{1}{i} + s(i))$, where $s : \mathbb{N} \to \mathbb{R}_{\geq 0}$ is a monotonically decreasing function; then,

$$\begin{aligned} \Delta(i) &= \frac{1}{n}\left(1 + i \cdot s(i) + ip\left(\frac{1}{i+1} + s(i+1)\right) - np\left(\frac{1}{i+1} + s(i+1)\right)\right) \\ &\geq \frac{1}{n}\left(1 + i \cdot s(i) - \frac{np}{i+1} - np \cdot s(i+1)\right) \end{aligned} \tag{4}$$

Suppose $s(1) = 0$; then, $\Delta(i) \geq \frac{1}{n}\left(1 - \frac{np}{1+i}\right)$. When $p = (2-c)/n$ for any $0 < c \leq 2$, $\Delta(i) \geq c/(2n)$ for all $i > 0$ (i.e., all $i \geq 1$), and hence the additive drift theorem yields an upper bound on the expected runtime of $\mathrm{MAHH_{OI}}$ of at most $g(n)/(c/(2n)) = O((n/c)\log n)$.

For $p \geq 2/n$, we will need to modify s to ensure that the drift $\Delta(i)$ remains positive throughout the search space. Observe that in equation (3), the bound on $\Delta(i)$ can be increased by increasing $s(i)$; and that as long as the expression in the parentheses produces at least a positive constant, a drift of $\Omega(1/n)$ can be maintained. First, we show that for $i \geq 2np$, no adjustment to $s(i)$ is necessary to achieve this:

$$\Delta(2np) \cdot n \geq 1 - \frac{np}{2np+1} \geq \frac{1}{2},$$

and as $\Delta(i)$ is an increasing function, $\Delta(i) \geq 1/(2n)$ for all $i \geq \lceil 2np \rceil$.

Let i_0 be the minimum Hamming distance to the optimum at which $1 - \frac{np}{i_0+1} \geq \frac{1}{2}$ and hence $\Delta(i) \geq 1/(2n)$ for all $i \geq i_0$. Per the above, $i_0 \leq \lceil 2np \rceil$. If $i_0 = 1$, a direct application of the additive drift theorem with $s(i) = 0$, as described previously, yields a $O(n \log n)$ upper bound on the runtime. Thus, it remains to consider the case that $2 \leq i_0 \leq \lceil 2np \rceil$.

Let $i_j = i_0 - j$ for $j \in \{1, \dots, i_0 - 1\}$. Using that $s(i_0) = 0$, we can adjust $s(i_1)$ to ensure that $\Delta(i_1)$ is sufficiently large. We will show by induction that $s(i_j) = (np)^j/(3i_j)$ is sufficient to ensure that $\Delta(i_j) \geq 1/(2n)$.

To begin, consider the drift at i_1:

$$\Delta(i_1) \cdot n \geq 1 + i_1 s(i_1) - \frac{np}{i_1+1} \geq 1 - \frac{np}{i_0+1} - \frac{np}{3} + i_1 s(i_1) \geq \frac{1}{2} - \frac{np}{3} + i_1 s(i_1),$$

using that $\frac{1}{i_j+1} = \frac{1}{i_0-j+1} = \frac{1}{i_0+1} + \frac{j}{i_0{}^2+(2-j)i_0+1} \leq \frac{1}{i_0+1} + \frac{1}{3}$ as $i_0 \geq 2$ and $j \leq i_0 - 1$. Hence, setting $s(i_1) \geq np/(3i_1)$ ensures that $\Delta(i_1) \geq 1/(2n)$, establishing the induction base case.

For $j \geq 2$, both $s(i_j)$ and $s(i_j + 1) = s(i_{j-1})$ are positive. Given that the induction hypothesis holds for $s(i_{j-1})$,

$$\begin{aligned}\Delta(i_j) \cdot n &\geq 1 + i_j s(i_j) - \frac{np}{i_j+1} - np\, s(i_j+1) \geq 1 - \frac{np}{i_0+1} - \frac{np}{3} - np\, s(i_{j-1}) + i_j s(i_j) \\ &\geq \frac{1}{2} - \frac{np}{3} - \frac{(np)^j}{3i_{j-1}} + i_j s(i_j) \geq \frac{1}{2} - \frac{(np)^j}{3} + i_j s(i_j),\end{aligned}$$

using that $\frac{np}{3} \leq \frac{(np)^2}{6}$ as $np \geq 2$, $i_{j-1} \geq 2$, and $j \geq 2$. Hence, setting $s(i_j) \geq (np)^j/(3i_j)$ ensures that $\Delta(i_j) \geq 1/(2n)$.

Thus, $s(i_j) = (np)^j/(3i_j) = (np)^{i_0-i_j}/(3i_j)$ for $1 \leq i_j < i_0 \leq \lceil 2np \rceil$, and $s(i) = 0$ for $i \geq i_0$ is sufficient to ensure that $\Delta(i) \geq 1/(2n)$ for all $i \geq 1$. Additionally, it holds that $s(1) \leq (np)^{i_0-1}/3 \leq (np)^{2np}/3$, and for $p = O((\log\log n)^{1-\epsilon}/n)$ with any constant $\epsilon > 0$, $(np)^{2np} = o(\log n)$, and hence $s(1) = o(\log n)$.

For $p = O((\log\log n)^{1-\epsilon}/n)$ and $np \geq 2$, $\sum_{i=1}^{n} s(i) \leq \sum_{i=1}^{n} s(1)/2^{i-1} < 2s(1) = o(\log n)$, and hence $g(n) = O(\log n)$. Applying the additive drift theorem with $X_0 = O(\log n)$ and $\delta = \Omega(1/n)$ thus yields that $X_t = 0$ is found in expectation after at most $X_0/\delta = O(n \log n)$ steps. Hence, MAHH$_{\text{OI}}$ with any $p = O((\log\log n)^{1-\epsilon}/n)$ and constant $\epsilon > 0$ optimises OneMax in $O(n \log n)$ iterations in expectation.

For $p = O((\log n)^{1-\epsilon}/n)$, $(np)^{2np} = o(n)$, and hence $g(n) = o(n)$. Applying the additive drift theorem yields that MAHH$_{\text{OI}}$ with any $p = O((\log n)^{1-\epsilon}/n)$ and constant $\epsilon > 0$ optimises OneMax in $o(n^2 \log n)$ iterations in expectation. □

For simplicity throughout the remainder of the paper, we consider MAHH$_{\text{OI}}$ with $p = \frac{1}{(1+\varepsilon)n}$ for any constant $\epsilon > 0$, which still gives the desired $O(n \log n)$ runtime. This value of p allows MAHH$_{\text{OI}}$ to maintain a positive drift throughout the search space of OneMax, and removes the need for complicating $(\log n)^{1-\varepsilon}/n$ terms in the subsequent calculations. We will show how with this parameter value the hyper-heuristics can, as well as hill-climbing efficiently, escape from difficult local optima effectively.

Regarding the performance of Metropolis for OneMax, we present the following Theorem from Jansen and Wegener [38], which provides a polynomial upper bound on the expected runtime of Metropolis for OneMax given that the function $\alpha(n)$ is chosen to be sufficiently large with respect to the problem size.

Theorem 8 *([38, Theorems 4&7]). The expected runtime of* Metropolis *for* OneMax *is polynomially bounded if and only if* $\alpha(n) = \Omega(n/\log n)$. *Furthermore, if* $\alpha(n) \geq \varepsilon n$ *for a constant* $\varepsilon > 0$, *then the expected runtime of* Metropolis *for* OneMax *is* $O(n \log n)$.

4. Multimodal optimisation: easy basins of attraction

We now analyse the performance of the search heuristics for the multimodal Cliff$_d$ $(1 < d < n/2)$ class of benchmark functions. Recall that Cliff$_d$ features local optima that the heuristics must escape before climbing a fitness gradient towards the global optimum. We refer to the local optimum at $i = n - d$ as the 'cliff', the two OneMax style hillclimbs as the 'first slope' and 'second slope' respectively (see Fig. 1a), and use d to denote the length of the cliff.

To find the global optimum of the Cliff$_d$ function, it is necessary to escape the local optimum, either by dropping down from the cliff and accepting a worse candidate solution and then climbing up the second slope, or by making a prohibitive jump to the global optimum on the other side of the cliff (this is possible with standard bit mutation, by requiring expected exponential time in the distance between the cliff and the optimum, but not with local mutations).

We now consider the performance of MAHH$_{\text{OI}}$ for Cliff$_d$. Clearly, with $p = 1$, MAHH$_{\text{OI}}$ reduces to a random walk across the fitness landscape. Similarly, if $p = 0$, with probability at least 1/2, the bit-string is initialised with at most $n/2$ 1-bits, and will hillclimb to the top of the cliff. There is no improving step from this position and the global optimum cannot be reached. By the law of total expectation, the expected runtime will be infinite. We will show that using MAHH$_{\text{OI}}$ with $p = \frac{1}{(1+\varepsilon)n}$, which has been shown to hillclimb efficiently (Theorem 7), will still allow worsening moves with sufficiently high probability to be able to move down from the cliff and reach the global optimum in expected polynomial time.

Theorem 10 bounds the expected runtime of MAHH$_{\text{OI}}$ for Cliff$_d$ from above. We begin, however, by introducing a helper Lemma which was proved by Droste et al. for trajectory based algorithms which can only change the number of 1-bits in the bit-string by 1 [29]. The lemma was subsequently used to analyse the performance of the $(1+1)$ EA for *noisy* OneMax for small noise strength [28]. Unlike in noisy optimisation, where the noise represents uncertainty with the respect to the true fitness of solutions, in hyper-heuristics the AM operator is intended to be helpful to the optimisation process by allowing the algorithm to escape from local optima.

Lemma 9 *([29, Lemma 3]). Let $E(T_i^+)$ be the expected time to reach a state with $i+1$ 1-bits, given a state with i 1-bits, and p_i^+ and p_i^- be the transition probabilities to reach a state with, respectively, $i+1$ and $i-1$ 1-bits. Then:*

$$E(T_i^+) = \frac{1}{p_i^+} + \frac{p_i^-}{p_i^+} \cdot E(T_{i-1}^+).$$

Within the context of non-elitist local search algorithms, such as MAHH$_{\text{OI}}$, the transition probability p_i^+ (p_i^-) refers to the probability of making a local mutation which increases (respectively decreases) the number of 1-bits in the offspring solution and accepting that solution.

Theorem 10. *The expected runtime of MAHH$_{OI}$ for* Cliff$_d$, *with $p = \frac{1}{(1+\varepsilon)n}$ for any constant $\varepsilon > 0$, is $O\left(n \log n + \frac{n^3}{d^2}\right)$.*

Proof. Let i denote the number of 1-bits in the bit-string at any time $t > 0$. We wish to bound the expected runtime from above by separately bounding four ‘stages’ of the optimisation process, each starting once all earlier stages have ended, and ending once a solution with at least certain number of 1-bits has been constructed for the first time during the optimisation process (i.e., the HH may go backwards afterwards yet will remain in the same stage): the first stage ends when $i \geq n-d$ is reached, the second when $i \geq n-d+1$, the third when $i \geq n-d+2$, and the fourth when $i = n$, i.e., the optimum has been constructed. We use $T_1, \ldots, T_4$ to denote the number of iterations the algorithm spends in each of these stages, and note that by definition of these stages, the number of iterations T before the optimum is reached is $T = T_1 + T_2 + T_3 + T_4$, and by linearity of expectation,

$$E(T) = E(T_1) + E(T_2) + E(T_3) + E(T_4). \tag{5}$$

In the first stage, while $i < n-d$, Cliff$_d$ resembles the OneMax function, and we can use the upper bound for the expected runtime of MAHH$_{\text{OI}}$ for OneMax with $p = \frac{1}{(1+\varepsilon)n}$ from Theorem 7. Hence, $E(T_1) = O(n \log n)$.

The second stage begins when $i \geq n-d$ for the first time, and ends when $i \geq n-d+1$ for the first time. When $i = n-d$, there are no improving moves. If the OI operator is selected, there will be no change in the candidate solution. Hence, any move must come from use of the AM operator, which is selected with probability $\frac{1}{(1+\varepsilon)n}$. A mutation step may either increase the number of 1-bits in the bit-string with probability d/n, or decrease the number of 1-bits with probability $(n-d)/n$. We use Lemma 9 to bound $E(T_2)$, the expected time to jump down from the cliff,

$$E(T_2) = E(T_{n-d}^+) = \frac{n^2(1+\varepsilon)}{d} + \frac{n-d}{d} \cdot E(T_{n-d-1}^+). \tag{6}$$

We now must bound $E(T_{n-d-1}^+)$. At $i = n-d-1$, the drift is as follows:

$$\begin{aligned}\Delta(n-d-1) &= \frac{d+1}{n} - \frac{1}{(1+\varepsilon)n} \cdot \frac{n-d-1}{n} \\ &= \frac{n(d+\varepsilon(d+1)) + d + 1}{n^2(1+\varepsilon)},\end{aligned}$$

as flipping any one of the $(d+1)$ remaining 0-bits increases the fitness by 1 and is accepted by either operator, and flipping any one of the $(n-d-1)$ 1-bits decreases the fitness by 1 and is only accepted if the AllMoves operator is chosen, which occurs with probability $p = 1/((1+\varepsilon)n)$.

Clearly, for all $0 \leq i \leq n-d-1$, the drift is bounded from above by the drift at $i = n-d-1$, while the distance to the required point from $i = n-d-1$ is 1. By the Additive Drift Theorem (Theorem 1), we have

$$E(T^{+}_{n-d-1}) \le 1/\Delta(n-d-1)$$
$$= \frac{n^2(1+\varepsilon)}{n(d+\varepsilon(d+1))+d+1} = O\left(\frac{n}{d}\right).$$

We can now return to bounding $E(T_2)$ in Equation (6):

$$E(T_2) = \frac{n^2(1+\varepsilon)}{d} + \frac{n-d}{d} \cdot E(T^{+}_{n-d-1})$$
$$= \frac{n^2(1+\varepsilon)}{d} + \frac{n-d}{d} \cdot O\left(\frac{n}{d}\right) = O\left(\frac{n^2}{d}\right). \quad (7)$$

The third stage begins with $i \ge n-d+1$, and ends once $i \ge n-d+2$. When $i = n-d+1$, all moves are improving moves and all moves will be accepted, regardless of the choice of the OI or AM operator. With probability $(d-1)/n$, the accepted move decreases the number of 1-bits to $i = n-d$ i.e., the hyper-heuristic returns to the local optimum. With probability $(n-d+1)/n$, the accepted move instead increases the number of 1-bits to $i = n-d+2$. We again use Lemma 9 to bound $E(T_3)$, the expected time to take one step up the second slope. Noting that $E(T^{+}_{n-d}) = O(n^2/d)$ by Equation (7), we get

$$E(T_3) = E(T^{+}_{n-d+1}) = \frac{n}{d-1} + \frac{n-d+1}{d-1} \cdot E(T^{+}_{n-d})$$
$$= \frac{n}{d-1} + \frac{n-d+1}{d-1} \cdot O\left(\frac{n^2}{d}\right) = O\left(\frac{n^3}{d^2}\right).$$

If $d = 2$, the CLIFF$_d$ function will have been optimised at this point. However, for $d \ge 3$, it is necessary to climb further up the second slope. If $d = 3$, we point out that $E(T_4) = E(T^{+}_{n-d+2})$, and by applying Lemma 9 bound

$$E(T^{+}_{n-d+2}) = \frac{n}{d-2} + \frac{1}{(1+\varepsilon)n} \cdot \frac{n-d+2}{d-2} \cdot E(T^{+}_{n-d+1})$$
$$= \frac{n}{d-2} + \frac{1}{(1+\varepsilon)n} \cdot \frac{n-d+2}{d-2} \cdot O\left(\frac{n^3}{d^2}\right)$$
$$= \frac{n}{d-2} + O\left(\frac{n^3}{d^3}\right).$$

For $d > 3$, we know that $i = n-d+2$ and $i = n-d+3$ are points on the second slope, and by applying Lemma 9 bound

$$E(T^{+}_{n-d+3}) = \frac{n}{d-3} + \frac{1}{(1+\varepsilon)n} \cdot \frac{n-d+3}{d-3} \cdot E(T^{+}_{n-d+2})$$
$$= \frac{n}{d-3} + \frac{1}{(1+\varepsilon)n} \cdot \frac{n-d+3}{d-3} \cdot O\left(\frac{n^3}{d^3}\right)$$
$$= \frac{n}{d-3} + O\left(\frac{n^3}{d^4}\right).$$

For $d > 4$ this trend will continue as MAHH$_{\text{OI}}$ progresses closer towards the global optimum; in particular, for $k < d$, we will have $E(T^{+}_{n-d+k}) = \frac{n}{d-k} + O\left(\frac{n^3}{d^k}\right)$.

Hence, $E(T_4) = \sum_{k=2}^{d-1} E(T^{+}_{n-d+k})$. The terms in the summation will be asymptotically dominated by the $O(n^3/d^3)$ term in $E(T^{+}_{n-d+2})$ if d is sub-linear, giving

$$E(T_4 \mid d = o(n)) \le d \cdot O(n^3/d^3) = O(n^3/d^2).$$

However, if d is linear in the problem size, the first terms will dominate, and we will have:

$$E(T_4 \mid d = \Theta(n)) = O(1) + \sum_{i=2}^{d-1} \frac{n}{d-i} \le O(1) + \sum_{i=0}^{d-1} \frac{n}{d-i}$$
$$= O(1) + n \cdot \sum_{j=1}^{d} \frac{1}{j} = O(n \log n).$$

Combining the bounds for the sub-linear and linear d, we conclude that

$$E(T_4) = O\left(n \log n + \frac{n^3}{d^2}\right).$$

We now return to our overall runtime bound from Equation (5) and complete the proof:

$$E(T) = E(T_1) + E(T_2) + E(T_3) + E(T_4)$$
$$= O\left(n\log n + \frac{n^2}{d} + \frac{n^3}{d^2} + n\log n + \frac{n^3}{d^2}\right) = O\left(n\log n + \frac{n^3}{d^2}\right). \quad \square$$

Theorem 10 gives an expected runtime bound of MAHH$_{\mathrm{OI}}$ for CLIFF$_d$ of $O\left(n\log n + n^3/d^2\right)$. This bound is smallest when d is large, suggesting that the algorithm is fastest when the cliff is hardest for elitist algorithms, i.e., a linear cliff length, $d = \Theta(n)$, gives an expected runtime of $O(n\log n)$. This runtime asymptotically matches the best case expected performance of artificial immune systems, which escape the local optimum with an ageing operator, also $O(n\log n)$ in expectation [8, 9,11,12,14], which is the best runtime known for hard CLIFF$_d$ functions. We suspect MAHH$_{\mathrm{OI}}$ is faster in practice (i.e., by having smaller leading constants in its expected runtime), but leave this proof for future work. Mutation based EAs have an expected runtime of $\Theta(n^d)$ for $d \le n/2$ [57]; for $d = \omega(1)$, this will give at least super-polynomial expected runtime in the length of the gap. Steady-state Genetic Algorithms which use crossover have recently been proven to be faster by at least a linear factor [17], but would still require exponential expected runtimes for large gaps.

The worst-case scenario for MAHH$_{\mathrm{OI}}$ is a constant gap length, giving a runtime of $O(n^3)$. This means that the (1 + 1) EA will outperform MAHH$_{\mathrm{OI}}$ if $d < 3$, but will be slower for any $3 < d \le n/2$, with a performance gap that increases exponentially with the distance between the cliff and the optimum.

Concerning non-elitist search heuristics, apart from the artificial immune systems, an upper bound of $O(n^\eta)$ (for $\eta \approx 3.9767$) on the expected runtime of the (1, λ) EA has recently been proved [35], for a carefully chosen parameter λ that satisfies $\log_{\frac{e}{e-1}} n \le \lambda = O(\log n)$, improving considerably upon the previously best known bound of n^{25}) [36]). The best case expected runtime of the non-elitist, strong-selection weak-mutation (SSWM) evolutionary regime, which rejects fitness-improving mutations with a non-zero probability, is at most $n^d/e^{\Omega(d)}$ [57]. Hence, MAHH$_{\mathrm{OI}}$ is also faster than SSWM for $3 < d \le n/2$.

We now compare the performance of MAHH$_{\mathrm{OI}}$ for CLIFF$_d$ with the non-elitist METROPOLIS algorithm (Algorithm 2). Theorem 11 shows that MAHH$_{\mathrm{OI}}$ will considerably outperform METROPOLIS for CLIFF$_d$ for all $d > 3$.

Theorem 11. *The expected runtime of* METROPOLIS *for* CLIFF$_d$ *is at least* $\min\left\{\frac{n-d+1}{2(d-1)} \cdot \left(\frac{cn}{\log n}\right)^{d-3/2}, n^{\omega(1)}\right\}$ *for some constant* $c > 0$.

Proof. Let i denote the number of 1-bits in the bit-string at any time $t > 0$. The CLIFF$_d$ function has two ONEMAX slopes ($0 \le i \le n-d$, $n-d+1 \le i \le n-1$). In order to reach the global optimum, it is necessary for METROPOLIS to optimise these two slopes. The expected runtime of METROPOLIS for ONEMAX is polynomially bounded in the problem size if and only if $\alpha(n) = \Omega(n/\log n)$ (by Theorem 8). Hence, $\alpha(n) = \Omega(n/\log n)$ is necessary in order to optimise the two slopes in polynomial time. However, the jump down to the bottom of the cliff will be difficult for METROPOLIS.

The randomly initialised candidate solution has at most $n/2$ bits with probability at least 1/2. METROPOLIS accepts the jump down from $i = n-d$ to $i = n-d+1$ with probability at most $\alpha(n)^{(n-d+3/2)-(n-d)} = \alpha(n)^{3/2-d}$. Hence, the expected time for this event to occur is at least $E(T^+_{n-d}) \ge \alpha(n)^{d-3/2}$. From this state, it is necessary to make at least one improving move to find the global optimum. Given that, at a state with $i = n-d+1$ 1-bits, all moves are accepted, we use Lemma 9, with $p^+_{n-d+1} = \frac{d-1}{n}$ and $p^-_{n-d+1} = \frac{n-d+1}{n}$:

$$E(T^+_{n-d+1}) = \frac{n}{d-1} + \frac{n-d+1}{d-1} \cdot E(T^+_{n-d})$$
$$\ge \frac{n}{d-1} + \frac{n-d+1}{d-1} \cdot \alpha(n)^{d-3/2}$$
$$\ge \left(\frac{n-d+1}{d-1} \cdot \alpha(n)^{d-3/2}\right).$$

Given that we want to minimise the overall runtime, we have $\alpha(n) = \Omega(n/\log n)$ such that the hillclimb can be done in polynomial time. However, the jump down the cliff will take

$$E(T^+_{n-d+1}) \ge \frac{n-d+1}{d-1} \cdot \left(\frac{cn}{\log n}\right)^{d-3/2}$$

for some constant $c > 0$.

We note that if $d = \omega(1)$, the term $E(T^+_{n-d+1})$ is exponential in the problem size, and setting $\alpha = \Omega(n/\log n)$ might not be optimal. In this case, the lower bound will be some exponential term, $n^{\omega(1)}$, i.e., the time taken for the two hillclimbs.

Overall, in order to optimise the function, the jump down the cliff has to be made (with probability at least 1/2) and at least one step up the second slope must be made. Hence,

$$E(T) \geq \min\left\{\frac{1}{2}\cdot\frac{n-d+1}{d-1}\cdot\left(\frac{cn}{\log n}\right)^{d-3/2}, n^{\omega(1)}\right\}. \quad \square$$

We have proven that while Metropolis cannot optimise hard Cliff_d variants in expected polynomial time, MAHH_{OI} is extremely efficient for hard cliffs, and has an expected runtime of $O(n^3)$ in the worst case.

5. Multimodal optimisation: hard basins of attraction

We now consider the Jump_m function class ($1 < m < n/2$) as an example of a multimodal function where MAHH_{OI} has a harder time escaping the local optima. Unlike Cliff_d, the fitness decreases on the second slope, making it harder to traverse.

Typically to optimise the Jump_m function (Fig. 1b), a search heuristic first reaches the local optimum at the top of the slope. Then, either a jump to the global optimum is made, which requires exponential expected time in the length of the jump for unbiased mutation operators, or a jump is made down to the slope of decreasing fitness, which must be traversed before the global optimum is found.

Theorem 12. *The runtime of MAHH_{OI} for Jump_m, with $p = \frac{1}{(1+\varepsilon)n}$, is at least $\Omega(n\log n + 2^{cm})$ for some constant $c > 0$ and any constant $\varepsilon > 0$, with probability at least $1 - 2^{-\Omega(m)}$.*

Proof. Let i represent the number of 1-bits in the bit-string at any time $t > 0$. First, we consider $m = O(\log n)$. The initialised candidate solution has at most $n/2$ bits with probability at least $1/2$. The expected time to optimise the function will be bounded from below by the expected time to reach the local optimum at the top of the slope. If $m = O(\log n)$, the algorithm must at least hillclimb to some point $i = n - k\ln n$, for some constant $k > 0$. The expected time to flip $n/2 - k\ln n$ bits correctly is $\Omega(n\log n)$; this can be proved by reusing arguments from the proof of [30, Lemma 10].

Similarly, the time to optimise the function will be bounded from below by the time to traverse the slope of decreasing fitness and find the global optimum, that is, to traverse the region where $n - m + 1 < i \leq n - 1$. Consider the negative drift on the number of 1-bits in the bit-string within this region:

$$\begin{aligned}
-\Delta(i) &= \frac{i}{n} - \frac{1}{(1+\varepsilon)n}\cdot\frac{n-i}{n} = \frac{i(n(1+\varepsilon)+1)-n}{n^2(1+\varepsilon)} \\
&\geq \frac{n/2(n(1+\varepsilon)+1)-n}{n^2(1+\varepsilon)} \\
&= \frac{n(1+\varepsilon)-1}{2n(1+\varepsilon)} = \frac{1}{2} - \frac{1}{2n(1+\varepsilon)} \geq 0.4.
\end{aligned}$$

We also note that the number of 1-bits can only change by 1 in each iteration, so there is no chance of escaping the area of negative drift with large jumps. We can apply the Negative Drift Theorem (Theorem 2) for the region from $i = n - m + 1$ to $i = n - 1$, i.e. of length $m - 2$. Hence there exists a constant $c' > 0$ such that, with probability at least $1 - 2^{-\Omega(m-2)} = 1 - 2^{-\Omega(m)}$, the global optimum is not found from the area of negative drift within $2^{c'\cdot(m-2)} = 2^{cm}$ steps, where $c = c'(m-2)/m > 0$ is a different constant. If $m = \omega(\log n)$, this term will dominate.

By considering both possibilities, we state that, for some constant $c > 0$, the expected time for MAHH_{OI} to optimise Jump_m is $\Omega(n\log n + 2^{cm})$ with probability at least $1 - 2^{-\Omega(m)}$. $\square$

The following theorem provides an upper bound on the expected runtime by considering the expected time for MAHH_{OI} to accept m consecutive worsening moves from the local to the global optimum.

Theorem 13. *The expected runtime of MAHH_{OI} for Jump_m, with $p = \frac{1}{(1+\varepsilon)n}$ for any constant $\varepsilon > 0$, is*

$$O\left(n\log n + \frac{(1+\varepsilon)^{m-1}n^{2m}}{m^2 m!}\right).$$

Proof. Let i denote the number of 1-bits in the bit-string at any time $t > 0$. The expected time to reach a bit-string with $i = n - m$ 1-bits is $O(n\log n)$, by Theorem 7.

From this position, in order to make a move towards the global optimum, it is necessary to flip one of the remaining 0-bits, and accept the result (with probability $p = \frac{1}{(1+\varepsilon)n}$). Starting from a search point with $i = n - m$ 1-bits (hence, m 0-bits), the probability of reaching a bit string with $i = n - 1$ 1-bits within the next $m - 1$ steps is given by

$$\prod_{i=2}^{m}\left(\frac{i}{n}\cdot\frac{1}{(1+\varepsilon)n}\right) = \frac{m!}{(1+\varepsilon)^{m-1}n^{2(m-1)}}.$$

Let $p^+_{n-m} \geq \frac{m!}{(1+\varepsilon)^{m-1}n^{2(m-1)}}$ be the probability that a bit-string with $n-1$ 1-bits is reached in $m-1$ consecutive steps from the local optimum at $i=n-m$, and p^-_{n-m} be the probability that a mutation which decreases the fitness to $i=n-m-1$ is accepted. By noting that $E(T^+_{n-m-1}) = O(n/m)$ (by the Additive Drift Theorem (Theorem 1)), we can use Lemma 9 to bound the expected time to reach such a bit string from the local optimum:

$$\begin{aligned} E(T^{opt}_{n-m}) &\leq \frac{(1+\varepsilon)^{m-1}n^{2(m-1)}}{m!} + \frac{(1+\varepsilon)^{m-1}n^{2(m-1)}}{m!} \cdot O\left(\frac{n}{m}\right) \\ &= O\left(\frac{(1+\varepsilon)^{m-1}n^{2m-1}}{m^2\cdot(m-1)!}\right). \end{aligned}$$

Hence, by a further application of Lemma 9, the expected time to reach the local optimum and traverse the slope of decreasing fitness to find the global optimum, is

$$E(T) = O\left(n\log n + \frac{(1+\varepsilon)^{m-1}n^{2m}}{m^2\cdot m!}\right). \quad \square$$

Using global mutations to jump allows smaller upper bounds on the expected runtime. The expected runtime of the (1+1) EA with mutation rate $1/n$ for Jump$_m$ is $\Theta(n^m)$ [30] and a bound of $O(n^{m-1})$ for Steady State $(\mu+1)$ Genetic Algorithms using crossover and a slightly higher mutation rate has recently been proved by Dang et al. [17], both outperforming our upper bound for MAHH$_{\text{OI}}$. Recent work has shown that by increasing even further the mutation rate, exponential speedups are achieved both by a static and adaptive Artificial Immune System [9,12,13] using hyper-mutations and by (1+1) EAs with heavy-tailed mutation operators following a power law distribution [31,24] or by using a stagnation detection mechanism to identify the best mutation rate to overcome the local optima [61,63,62] (however, the expected runtime is still exponential in m). A compact GA has super-polynomial speedups over these algorithms for super-constant jump lengths when $m=o(n)$ [33]. In particular for logarithmic jump lengths the algorithm optimises Jump$_m$ in expected polynomial time. We point out that steady state GAs with diversity mechanisms that enhance the power of crossover can optimise Jump$_m$ in expected time $O(mn\log n + 4^m)$ for jumps up to $m=n/8$) [19] and in $\Theta(n\log n + 4^m)$ with an unrealistically small crossover probability of $O(m/n)$ [41].

Concerning algorithms that use non-elitism to escape from local optima, while the (μ,λ) EA does not improve over the expected runtime of the elitist $(\mu+\lambda)$ EA [20], we will now show that Metropolis has worse performance than MAHH$_{\text{OI}}$ on Jump$_m$. The large fitness difference between the two points at $i=n-m$ and $i=n-m+1$ makes Metropolis unlikely to accept the $i=n-m+1$ point when $\alpha(n)$ is set high enough to enable efficient hill-climbing of the first slope.

Theorem 14. *The runtime of* Metropolis *for* Jump$_m$ *is* $2^{\Omega(n)}$*, with probability at least* $1-2^{-\Omega(n)}$.

Proof. Let i represent the number of 1-bits in the bit-string at any time $t>0$. The initialised candidate solution has at most $n/2$ 1-bits with probability at least 1/2. The jump down from the local optimum will be accepted with probability $\alpha(n)^{f(n-m+1)-f(n-m)} = \alpha(n)^{m-1-n}$, and will take expected time at least $E(T^+_{n-m}) \geq \alpha(n)^{n+1-m}$.

For $i>n-m$, we can again use Lemma 9, with $p^+_i = \alpha(n)^{-1}\cdot\frac{n-i}{n}$ and $p^-_i = \frac{i}{n}$, to bound the expected time to transition between neighbouring states:

$$\begin{aligned} E(T^+_{n-m+1}) &= \alpha(n)\cdot\frac{n}{m-1} + \alpha(n)\cdot\frac{n-m+1}{m-1}\cdot E(T^+_{n-m}) \\ &\geq \alpha(n)\cdot\frac{n}{m-1} + \alpha(n)\cdot\frac{n-m+1}{m-1}\cdot\alpha(n)^{n+1-m} \\ &\geq \alpha(n)^{n+2-m} \\ E(T^+_{n-m+2}) &= \alpha(n)\left(\frac{n}{m-2} + \frac{n-m+2}{m-2}\cdot E(T^+_{n-m+1})\right) \\ &\geq \alpha(n)\cdot\frac{n}{m-2} + \alpha(n)\cdot\frac{n-m+2}{m-2}\cdot\alpha(n)^{n+2-m} \\ &\geq \alpha(n)^{n+3-m}. \end{aligned}$$

We note that, in general, $E(T^+_{n-m+k}) \geq \alpha(n)^{n+(k+1)-m}$, and hence

$$E(T) \geq E(T^+_{n-2}) \geq \alpha(n)^{n+(m-2+1)-m} = \alpha(n)^{n-1}.$$

We now require a lower bound on $\alpha(n)$. If $m>n/5$, consider the negative drift in the number of 1-bits in the region from $i=n-m+1$ to $i=n-1$. Clearly, $\alpha(n)=1$ is the best choice for this region, yet will still lead to a large area of negative drift:

$$-\Delta(i) = \frac{i}{n} - \frac{n-i}{n} \geq -\Delta\left(\frac{4n}{5}\right) = \frac{3}{5}.$$

Since Metropolis is a local search algorithm, this region of negative drift cannot be escaped by a large jump. Hence, by the Negative Drift Theorem (Theorem 2), the time to escape this region will be at least $2^{\Omega(n)}$ with overwhelming probability $1-2^{-\Omega(n)}$. This will also be a lower bound on the time to optimise the function.

If $m \leq n/5$, with probability at least $1/2$, Metropolis must hillclimb from $i = 3n/4$ to $i = 4n/5$. Consider the drift on the fitness in this region when $\alpha(n) = 2$:

$$-\Delta(i) = \frac{1}{2} \cdot \frac{i}{n} - \frac{n-i}{n} \geq -\Delta\left(\frac{3n}{4}\right) = \frac{1}{8}.$$

Clearly, to avoid a large region of negative drift, $\alpha(n) \geq 2$ is required; giving $E(T) \geq 2^{n-1} = 2^{\Omega(n)}$. □

Unlike Metropolis, MAHH$_{\text{OI}}$ does not consider the magnitude of the fitness difference between two solutions to decide whether to accept worsenings. This allows it to optimise Jump$_m$ with smaller jump sizes, such as $m = \Theta(1)$, in expected polynomial time, while Metropolis requires exponential time in expectation in all cases.

6. Multimodal optimisation: basins of attraction of tuneable gradient and size

The CliffJump$_{d,r,s}$ function class (see Fig. 1c) combines elements of both the Cliff$_d$ and Jump$_m$ function classes. Upon reaching the local optimum (i.e., the top of the first OneMax slope), r fitness-worsening mutations, each decreasing fitness of the current solution by s, must be overcome before the slope leading to the global optimum can be reached. For algorithms that flip one bit uniformly at random in each mutation step (e.g., RLS), setting $r = 1$ makes CliffJump$_{d,r,s}$ similar to Cliff$_d$, while setting $r = d$ makes it similar to Jump$_m$; thus, we consider this function class with parameters $1 < r < d < n/2$.

To begin, the following theorem shows that if the radius of the region of attraction around the local optimum is too large i.e., super-logarithmic, both MAHH$_{\text{OI}}$ and Metropolis require super-polynomial expected time to optimise CliffJump$_{d,r,s}$, just like elitist heuristics using standard bit mutation.

Theorem 15. *On* CliffJump$_{d,r,s}$ *with $r = \omega(\log n)$ and $d < (0.5 - c)n$ for any constant $0 < c < 0.5$ and any $s > 0$, the expected optimisation time of MAHH$_{OI}$ and* Metropolis*, with any parameters, is super-polynomial.*

Proof. We note that both algorithms are with probability $1/2$ initialised with a solution containing at most $n/2$ 1-bits, in which case they will need to descend the entire length of the negative-sloped segment of CliffJump$_{d,r,s}$. We will apply the Negative Drift Theorem to show that regardless of the choice of algorithm parameters, reaching a solution with $n-d+r$ 1-bits will then take super-polynomial time in expectation, and hence the overall expected runtime of either algorithm on CliffJump$_{d,r,s}$ will be super-polynomial by the law of total probability.

Let X_t denote the number of 1-bits in the current solution after t'th accepted mutation changing the number of 1-bits in the current solution. Suppose $n-d \leq X_t < n-d+r$. A solution with fewer 1-bits is generated with probability $X_t/n \geq (0.5+c)$ and accepted with probability 1; let $p^- \geq 0.5+c$ denote a lower bound on the probability that such a solution is generated and accepted in a single iteration. A solution with more 1-bits is generated with probability $(n-X_t)/n \leq 0.5-c$, and accepted with probability at most 1; let $p^+ := 0.5-c$ denote an upper bound on the probability that such a solution is generated and accepted in a single iteration. Thus, conditional on X_t changing, it increases by 1 with probability at most $p^+/(p^+ + p^-) \leq 0.5-c$, and decreases by 1 with probability at least $p^-/(p^+ + p^-) \geq 0.5+c$. Thus, the expected change in the number of 1-bits is:

$$E(X_{t+1} - X_t \mid n-d \leq X_t < n-d+r) \leq 1 \cdot (0.5-c) - 1 \cdot (0.5+c) = -2c = -\Theta(1).$$

As both algorithms employ local mutations, $P(|X_{t+1} - X_t| > 1) = 0$, and hence we can apply the Negative Drift Theorem (Theorem 2) to lower bound on the number of steps needed to increase X_t from $n-d$ to $n-d+r$ (a region of length $\Omega(r)$): with probability at least $1-2^{-\Omega(r)} = 1-2^{-\omega(\log n)}$, more than $2^{\Omega(r)} = 2^{\omega(\log n)} = n^{\omega(1)}$ accepted mutations changing the number of 1-bits are required. This is also a lower bound on the number of iterations of MAHH$_{\text{OI}}$ (or Metropolis) required to construct a solution with at least $n-d+r$ 1-bits starting with a solution with at most $n-d$ 1-bits. Thus, the expected optimisation time of MAHH$_{\text{OI}}$ and Metropolis, with any parameters, on CliffJump$_{d,r,s}$ with $r = \omega(\log n)$ and $d < (0.5-c)$ for any constant $0 < c < 0.5$ is super-polynomial. □

The following theorem shows that when the radius of the region of attraction around the local optimum is at most constant, MAHH$_{\text{OI}}$ is able to optimise CliffJump$_{d,r,s}$ in expected polynomial time.

Theorem 16. *The expected runtime of MAHH$_{OI}$, with $p = \frac{1}{(1+\varepsilon)n}$ for any constant $\varepsilon > 0$, for* CliffJump$_{d,r,s}$ *with $1 \leq r < d$ and any $s > 0$ is at most*

$$O\left((d-r)\frac{(1+\varepsilon)^r n^{2r+2}}{(r+1)^2(r+1)!}\right).$$

Proof. As MAHH$_{\mathrm{OI}}$ only considers the sign of the fitness difference between the parent and offspring solutions (and not the magnitude), its behaviour is unaffected by the choice of the s parameter of CLIFFJUMP$_{d,r,s}$.

For $r=1$, the result can be proven exactly as in the proof of Theorem 10. For $r>1$, we will adapt the proof of Theorem 10 to cope with the larger basin of attraction around the local optimum.

Let T_u be the number of iterations before MAHH$_{\mathrm{OI}}$ takes the first step up the slope leading to the global optimum for the first time (i.e. constructs a solution with ONEMAX $(x)=n-d+r+1$). This is at most as difficult as finding the global optimum of JUMP$_m$ for $m=r+1$, and thus $E(T_u)=O((1+\varepsilon)^r n^{2r+2}/((r+1)^2(r+1)!)$ per Theorem 13. If $d=r+1$, the global optimum has been found at this point; otherwise, we need to separately bound the time required to climb the second slope, taking into the account the possibility of MAHH$_{\mathrm{OI}}$ returning to the local optimum. In the language of the proof of Theorem 10, we have that $E(T_u)=E(T_1)+E(T_2)+E(T_3)$, and need to bound $E(T_4)$.

Let T_j^+ be the number of iterations between a solution with j 0-bits being accepted and the next time a solution with $(j-1)$ 0-bits is accepted. We bound $E(T_{d-r}^+)\le E(T_u)$, and for $1\le j<d-r$,

$$\begin{aligned}E(T_j^+)&=1+j/n\cdot 0+(1-j/n)\cdot(p\cdot E(T_{j+1}^+)+E(T_j^+))\\&=\frac{1+(1-j/n)\cdot p\cdot E(T_{j+1}^+)}{1-(1-j/n)}=\frac{n+(n-j)\cdot p\cdot E(T_{j+1}^+)}{j}=O(n+E(T_u))\end{aligned}$$

as $p=O(1/n)$. Hence,

$$E(T_4)=E\left(\sum_{j=1}^{d-r-1}T_j^+\right)=\sum_{j=1}^{d-r-1}E(T_j^+)\le(d-r-1)\cdot O(n+E(T_u)).$$

Thus, MAHH$_{\mathrm{OI}}$ finds the global optimum of CLIFFJUMP$_{d,r,s}$ after at most $O((d-r)(n\log n+((1+\varepsilon)^r n^{2r+2})/(r+1)^2(r+1)!))=O((d-r)((1+\varepsilon)^r n^{2r+2})/((r+1)^2(r+1)!))$ iterations in expectation. □

We can provide a tighter upper bound for MAHH$_{\mathrm{OI}}$ on CLIFFJUMP$_{d,r,s}$ when the negative slope is far away from the global optimum.

Theorem 17. *The expected runtime of MAHH$_{\mathrm{OI}}$, with $p=\frac{1}{(1+\varepsilon)n}$ for any constant $\varepsilon>0$, for* CLIFFJUMP$_{d,r,s}$ *with $d=\Theta(n)$, $r=o(n)$ and any $s>0$ is at most $O(n^{r+3})$.*

Proof. For $d=\Theta(n)$ and $r=o(n)$, the probability of mutating to a solution with a higher number of 1-bits on the negative slope section of CLIFFJUMP$_{d,r,s}$ is at least $(n-d+r)/n=\Theta(1)$.

A mutation which increases the number of 1-bits while on the negative slope is accepted with probability $p=\frac{1}{(1+\varepsilon)n}$ (since the move acceptance probability of MAHH$_{\mathrm{OI}}$ is fixed, and not affected by the fitness decrease s). Hence, from the local optimum, the probability of accepting r consecutive decreasing moves to reach the local minimum and then move one step up the next slope is at least

$$\left(\frac{1}{(1+\varepsilon)cn}\right)^{r+1},$$

for some constant $c>0$.

Let $p_{n-d}^+\ge\left(\frac{1}{(1+\varepsilon)cn}\right)^{r+1}$ be the probability that a solution with $i=n-d+r+1$ 1-bits is reached in $r+1$ consecutive steps from a solution with $i=n-d$ 1-bits. We can bound the time to reach a bit-string with $i=n-d$ 1-bits after random initialisation by $O(n)$ (see Theorem 7), and hence we can bound the time to reach a solution with $i=n-d+r+1$ 1-bits (using Lemma 9) by

$$O(n)\cdot O\left(n^{r+1}\right)=O\left(n^{r+2}\right).$$

Similarly, reusing arguments from the proof of Theorem 16, the expected time required to climb the second positive slope (of length $d-r=O(n)$) and reach the global optimum is bounded by

$$O(n\log n)+O(n)\cdot O(n^{r+2})=O(n^{r+3})$$

using Lemma 9. Hence, the overall expected runtime can be bounded by $O(n^{r+3})$. □

Theorem 16 and Theorem 17 respectively give upper bounds for MAHH$_{\text{OI}}$ on CLIFFJUMP$_{d,r,s}$ for the general case and the case where the local optimum is far away from the global optimum. The following corollary shows that the non-elitist MAHH$_{\text{OI}}$ can outperform the well-known elitist $(\mu + \lambda)$ EAs with standard bit mutation for CLIFFJUMP$_{d,r,s}$ in both cases, provided that s is at least a sufficiently large constant. The result generalises to any μ and λ since the populations do not provide any advantage over the $(1+1)$ EA on the JUMP$_m$ slope [20].

Corollary 18. *MAHH$_{OI}$ is faster than the $(1+1)$ EA in expectation for* CLIFFJUMP$_{d,r,s}$ *when $s > 1 + 3/r$. If $d = \Theta(n)$, then MAHH$_{OI}$ is faster than the $(1+1)$ EA in expectation when $s > 3/r$.*

Proof. The $(1+1)$ EA with standard bit mutation is an elitist algorithm, and will only accept mutations which increase the fitness of the solution. From the local optimum of CLIFFJUMP$_{d,r,s}$ (i.e., when the solution has $i = n - d$ 1-bits), the nearest solution with a higher fitness consists of $i = \lceil n - d + r + rs \rceil$ 1-bits. Hence, the expected time to perform such a mutation from the local optimum is $\Omega(n^{r(s+1)})$, which is an overall lower bound for the expected runtime of the $(1+1)$ EA.

Theorem 16 gives a general upper bound for MAHH$_{\text{OI}}$ on CLIFFJUMP$_{d,r,s}$ of $O((d-r)/r \cdot n^{2r+2}) = O(n^{2r+3})$. Hence, we can state that MAHH$_{\text{OI}}$ outperforms the $(1+1)$ EA (in expectation) when $2r + 3 < r(s+1)$. That is, when $s > 1 + 3/r$.

When $d = \Theta(n)$, Theorem 17 gives a tighter upper bound for MAHH$_{\text{OI}}$ on CLIFFJUMP$_{d,r,s}$ of $O(n^{r+3})$. Hence, we can state than MAHH$_{\text{OI}}$ outperforms the $(1+1)$ EA in this case (in expectation) when $r + 3 < r(s+1)$. That is, when $s > 3/r$. □

In particular, for minimal basins of attraction when $r = 1$, MAHH$_{\text{OI}}$ is faster than the $(1+1)$ EA for any $s > 4$ in the general case, and for $s > 3$ when $d = \Theta(n)$. Furthermore, for any basin of attraction of size $r \geq 3$, then MAHH$_{\text{OI}}$ is faster when $s > 2$ and $s > 1$ respectively. We further note that if CLIFFJUMP$_{d,r,s}$ was modified such that only the global optimum point had a fitness value exceeding that of the local optimum, then the expected runtime of the $(1+1)$ EA would be $\Omega(n^d)$. If $d = \Theta(n)$, then the expected runtime would be exponential in all cases, whereas MAHH$_{\text{OI}}$ can be polynomially bounded in such a case when r is a constant. Thus, the function illustrates how non-elitism may allow to efficiently escape from a wide range of local optima where elitist EAs are inefficient.

Regarding METROPOLIS, we first show that it is able to exactly match the performance of MAHH$_{\text{OI}}$ for CLIFFJUMP$_{d,r,s}$ when $s = 1$.

Theorem 19. *On* CLIFFJUMP$_{d,r,s}$ *with $s = 1$ and any integers d and r, MAHH$_{OI}$ and* METROPOLIS *can be configured to behave identically. That is, when $p = \alpha(n)^{-1}$, the two algorithms maintain identical state probability distributions throughout their runs.*

Proof. With the exception of the global optimum, the fitness difference on CLIFFJUMP$_{d,r,1}$ between any two bit-strings that are at a Hamming distance of 1 is either -1 or 1. Thus, any offspring except the global optimum differs from its parent in fitness by either 1 or -1.

If the offspring is better than its parent (either by a fitness difference of 1, or because the offspring is the global optimum), both MAHH$_{\text{OI}}$ and METROPOLIS will accept the offspring solution. If the offspring is worse than its parent (and the parent isn't the global optimum), MAHH$_{\text{OI}}$ will accept the offspring solution with probability p, and METROPOLIS will accept the offspring solution with probability $\alpha(n)^{-1}$.

Thus, $p = \alpha(n)^{-1}$ ensures that, until the global optimum is found, both algorithms have the same probability of accepting each offspring solution considered during the optimisation process. As both algorithms are initialised with a solution chosen uniformly at random, use the same mutation operator to generate offspring solutions, and consider exactly one offspring in each iteration, they therefore maintain identical probabilities of having observed the global optimum within any given iteration budget. □

We can extend this argument to show that if the fitness differences between any two adjacent points are equal across the entire search space, MAHH$_{\text{OI}}$ and METROPOLIS can be configured to have identical performance. This holds for all fitness functions which display such a property.

Finally, because METROPOLIS is sensitive to the magnitude of the fitness differences between neighbouring solutions, we show that there exist choices of the CLIFFJUMP$_{d,r,s}$ parameter s that allow METROPOLIS to outperform MAHH$_{\text{OI}}$, and choices that allow MAHH$_{\text{OI}}$ to outperform METROPOLIS.

Theorem 20. *On* CLIFFJUMP$_{d,r,s}$, *if $0 < s < 1$, $\alpha(n)$ can be chosen such that* METROPOLIS *outperforms MAHH$_{OI}$ with any choice of $0 < p < 1$. If $s > 1$, p can be chosen such that MAHH$_{OI}$ outperforms* METROPOLIS *with any choice of $\alpha(n)$.*

Proof. On CLIFFJUMP$_{d,r,s}$, there are two types of mutations which can reduce the fitness value of a solution: those that reduce the number of 1-bits while the solution is on the positive slopes (i.e., $|x|_1 \leq n - d$ or $|x|_1 > n - d + r$) and hence reduce fitness by 1, and those that increase the number of 1-bits while the solution is on the negative slope (i.e., $n - d \leq |x|_1 < n - d + r$) and hence reduce fitness by s. Recall that METROPOLIS accepts mutations which decrease fitness by Δf with probability $\alpha(n)^{-\Delta f}$. Thus, if $s < 1$, it is easier for METROPOLIS to accept mutations increasing the number of 1-bits while on the negative slope compared to mutations decreasing the number of 1-bits while on the positive slope, while the opposite holds if $s > 1$.

Suppose $s < 1$, and consider how the parameters of METROPOLIS can be chosen so as to outperform MAHH$_{\text{OI}}$ with a fixed parameter p. Choosing $\alpha(n) = 1/p$ ensures that METROPOLIS matches MAHH$_{\text{OI}}$'s probabilities of making progress while on the positive slopes, and accepts mutations that increase the number of 1-bits in the solution while on the negative slope with probability $\alpha(n)^{-s} = p^{-s} > p$. Thus, METROPOLIS with $\alpha(n) = p^{-1}$ is at least as fast on the positive slopes as MAHH$_{\text{OI}}$ with any p, while traversing the negative slope faster than MAHH$_{\text{OI}}$ with that p. Choosing $\alpha(n) = p^{-1/s}$ ensures that METROPOLIS matches MAHH$_{\text{OI}}$'s probabilities of making progress while on the negative slope, while also making METROPOLIS less likely to accept detrimental mutations while on the positive slopes. Choosing an $\alpha(n)$ between these values will result in (reduced) improvement over MAHH$_{\text{OI}}$ on both slope types. Combining these results yields that METROPOLIS with $p^{-1} \le \alpha(n) \le p^{-1/s}$ outperforms MAHH$_{\text{OI}}$ for CLIFFJUMP$_{d,r,s}$ for any parameter p, with $0 < s < 1$.

Similarly, $\alpha(n)^{-s} \le p \le \alpha(n)^{-1}$ ensures that MAHH$_{\text{OI}}$ outperforms METROPOLIS for CLIFFJUMP$_{d,r,s}$ with $s > 1$. Here, choosing $p = \alpha(n)^{-s}$ allows MAHH$_{\text{OI}}$ to replicate METROPOLIS behaviour on the negative slope while making MAHH$_{\text{OI}}$ less likely to accept worsenings on the positive slopes, while $p = \alpha(n)^{-1}$ allows MAHH$_{\text{OI}}$ to replicate METROPOLIS behaviour on the positive slopes while making MAHH$_{\text{OI}}$ more likely to accept mutations increasing the number of 1-bits while on the negative slope, with values between these two boundaries providing a mix of both behaviours. □

Theorem 20 implies that METROPOLIS is preferable for CLIFFJUMP$_{d,r,s}$ when the fitness differences between adjacent points on the negative slope are smaller in magnitude than between adjacent points on the positive slopes, while MAHH$_{\text{OI}}$ is preferable when the opposite is true.

MAHH$_{\text{OI}}$ is efficient for CLIFFJUMP$_{d,r,s}$ while the radius of the local optima's basin is constant, for any $s > 0$, as it is able to escape the jump and cliff sections in polynomial time. We now prove that METROPOLIS, however, is only efficient when the negative slope is gentle, short, and far away from the global optimum.

Theorem 21. *The expected runtime of* METROPOLIS *for* CLIFFJUMP$_{d,r,s}$ *with* $r \ge 1$ *is super-polynomial if* $s = \omega(1)$. *Furthermore, if* $d < (0.5 - c)n$ *for any constant* $0 < c < 0.5$, *the expected runtime is polynomially bounded if and only if* $r = O(\log n)$ *and* $rs = O(1)$.

Proof. Recall from Theorem 8 that the expected runtime of METROPOLIS is polynomially bounded on ONEMAX if and only if $\alpha(n) = \Omega(n/\log n)$, and this bound on $\alpha(n)$ must hold in order to guarantee polynomial expected runtime on the ONEMAX sections of CLIFFJUMP$_{d,r,s}$.

METROPOLIS is initialised with a solution containing at most $n/2 - 1$ 1-bits with probability $1/2$, in which case it is required to descend the entirety of the negative slope of CLIFFJUMP$_{d,r,s}$ before finding the global optimum. METROPOLIS accepts moves which increase the number of 1-bits on the negative slope with probability $\alpha(n)^{-s}$, and such a move will take at least $\alpha(n)^s$ expected time. Since $\alpha(n) = \Omega(n/\log n)$ is required for polynomial expected runtime in the two ONEMAX slopes, a bound of $s = O(1)$ is required for polynomial expected runtime in the negative slope (of length $r \ge 1$). Therefore, $s = O(1)$ is necessary in order to bound the overall expected runtime by a polynomial, and the setting $s = \omega(1)$ will give a super-polynomial expected runtime.

We know from Theorem 15 that for $r = \omega(\log n)$ and $d < (0.5 - c)n$ for some constant $0 < c < 0.5$, the expected runtime of METROPOLIS on CLIFFJUMP$_{d,r,s}$ is super-polynomial. Therefore, we consider the case where $r = O(\log n)$.

Let $E(T_{\text{LM}})$ denote the expected time required to reach the local maximum (i.e., the expected time to find a solution with $i = n - d$ 1-bits for the first time) after initialisation. By Lemma 9, the expected time to take one step down the negative slope is (for constants $c > 0$ and $c' > 0$):

$$c \cdot \alpha(n)^s + c'\alpha(n)^{s-1} \cdot E(T_{\text{LM}}) = \Theta(\alpha(n)^s),$$

since $E(T_{\text{LM}}) = O(n)$. Furthermore, by reusing arguments from the proof of Theorem 14 (i.e., the repeated application of Lemma 9), we can show that the expected time to take r fitness-decreasing steps down the negative slope is $\Theta(\alpha(n)^{rs})$.

Therefore, since $\alpha(n) = \Omega(n/\log n)$ is required for polynomial expected runtime in the two ONEMAX slopes, $rs = O(1)$ (and $r = O(\log n)$) is necessary to give a polynomial expected runtime for METROPOLIS on CLIFFJUMP$_{d,r,s}$ with these settings, and $rs = \omega(1)$ will give a super-polynomial expected runtime. □

7. When Metropolis outperforms the move acceptance hyper-heuristic

In the previous sections we analysed the performance of MAHH$_{\text{OI}}$ and METROPOLIS for several multimodal landscapes with typical characteristics. The analysis has showed how MAHH$_{\text{OI}}$ is more robust than METROPOLIS at escaping local optima since it is not sensitive to the fitness-gradients in the basins-of-attraction that have to be overcome. On the other hand METROPOLIS exhibits better performance on smooth basins of attraction exhibiting slow fitness-decreasing slopes.

In this section we present an analysis on the GENTLENEGATIVESLOPE (GNS) function, which was designed to show how METROPOLIS can take advantage of a very long, slowly decreasing slope that is surrounded by points of considerably lower fitness. Since METROPOLIS is unlikely to accept moves which take it away from the smooth fitness-decreasing gradient towards the optimum, and MAHH$_{\text{OI}}$ has a high chance to accept such disadvantageous moves, this function allows us to highlight landscape characteristics where Metropolis considerably outperforms MAHH$_{\text{OI}}$. In particular, MAHH$_{\text{OI}}$ is penalised by not taking fitness-differences into account when accepting worsening solutions, the same characteristic that was crucial for the HH to either outperform or be a preferable choice to Metropolis on all the fitness landscapes previously considered in this paper.

The following theorem proves that Metropolis can optimise GNS in expected polynomial time when using a parameter setting that allows it to both efficiently hill-climb the steep OneMax slope and perform a random walk across the ridge linking the local optimum to the global optimum due to a large fitness difference between the ridge and the rest of the search space.

Theorem 22 *([38, Theorem 11]). The expected runtime of* Metropolis, *with* $\alpha(n) = n^{4/2^n}$, *for* GNS *is* $O(n^3)$.

The Metropolis algorithm with $\alpha(n) = n^{4/2^n}$ will reach the local optimum at 1^n in $O(n \log n)$ expected iterations, before a random search along the path leads the algorithm to the global optimum at 0^n in expected $O(n^3)$ time [38]. Jansen and Wegener also proved that the $(1+1)$ EA has exponential expected runtime for GNS, i.e., at least $2^{\Omega(n)}$ for any choice of the mutation rate [38]. The initialised solution of the $(1+1)$ EA will have at least $n/3$ 1-bits with probability $1 - 2^{-\Omega(n)}$, and finding a path point 1^i0^{n-i} with $i \le n/6$ will take exponential time. If a path point 1^i0^{n-i} is sampled with $i > n/6$, a large jump is required to find the global optimum, taking exponential expected time.

The following theorem proves that MAHH$_{\text{OI}}$ also exhibits poor performance for GNS regardless of the choice of the parameter p. Essentially, since MAHH$_{\text{OI}}$ does not take fitness differences into account when choosing whether to accept fitness-decreasing offspring, it is not able to stay on and efficiently navigate the gentle ridge between the local optimum and the global optimum of GNS.

Theorem 23. *The runtime of MAHH$_{OI}$ for* GNS, *with any* $0 \le p \le 1$, *is at least* $n^{\omega(1)}$ *with probability at least* $1 - n^{-\omega(1)}$.

Proof. As GNS has a unique global optimum at 0^n, Theorem 6 applies, and the expected runtime of MAHH$_{\text{OI}}$ for GNS is at least as large as the expected runtime of MAHH$_{\text{OI}}$ on OneMax for the same p. For $p = \omega\left((\sqrt{n}\log n)/n\right)$, applying Theorem 5 thus yields that MAHH$_{\text{OI}}$ requires at least $n^{\omega(1)}$ iterations with probability at least $1 - n^{-\omega(1)}$ to optimise GNS.

Suppose $p < n^{3/4}/n$, and consider the behaviour of MAHH$_{\text{OI}}$ in the region where the OneMax value of MAHH$_{\text{OI}}$'s current solution is between $\sqrt[4]{n}/4$ and $\sqrt[4]{n}/2$. MAHH$_{\text{OI}}$ can only decrease the OneMax value of its current solution when it flips one of the remaining 1-bits and applies the AM selection operator, which occurs with probability $p \cdot \text{OneMax}(x)/n \le 1/(2n)$. Within this region, MAHH$_{\text{OI}}$ can *always* increase the OneMax value of its current solution when it flips the first 0-bit in the solution, which occurs with probability $1/n$. Thus, conditional on the OneMax value of the current solution changing in an iteration of MAHH$_{\text{OI}}$, it increases by 1 (as MAHH$_{\text{OI}}$ applies the RLS mutation operator) with probability at least $(1/n)/(1/n + 1/(2n)) \ge 2/3$, and decreases by 1 with probability at most $(1/(2n))/(1/n + 1/(2n)) \le 1/3$.

Let X_t be the OneMax value of MAHH$_{\text{OI}}$'s current solution after its t'th change. We focus on the region of length $\ell = \sqrt[4]{n}/4$ between $X_t \ge a := \sqrt[4]{n}/4$ and $X_t \le b := \sqrt[4]{n}/2$. Then,

$$E(X_{t+1} - X_t \mid \sqrt[4]{n}/4 \le X_t \le \sqrt[4]{n}/2) \ge 2/3 - 1/3 = 1/3.$$

Since MAHH$_{\text{OI}}$ applies RLS mutation, $\Pr(|X_{t+1} - X_t| > 1) = 0$. Thus, all the conditions to apply the Negative Drift Theorem (Theorem 2) are satisfied, and the first hitting time of $X_t \le a$ is at least $2^{\Omega(n^{1/4})}$ with probability at least $1 - 2^{-\Omega(n^{1/4})}$.

It remains to note that MAHH$_{\text{OI}}$ is initialised with at least $n/4$ 1-bits in the current solution with probability at least $1 - e^{-\Omega(n)}$. Thus, $X_0 > b$, and as the runtime of MAHH$_{\text{OI}}$ for GNS is at least the number of iterations before a solution with $X_t < a$ is first constructed during which the OneMax value of its solution changes, it holds that the runtime of MAHH$_{\text{OI}}$ with $p \le n^{3/4}/n$ for GNS is at least $2^{\Omega(n^{1/4})} \in n^{\omega(1)}$ with probability at least $1 - 2^{-\Omega(n^{1/4})} \in 1 - n^{-\omega(1)}$.

Finally, we note that $n^{3/4}/n \in \omega((\sqrt{n}\log n)/n)$. Therefore, regardless of what value p is set to, at least one of the two arguments above applies. Hence MAHH$_{\text{OI}}$ requires, with probability at least $1 - n^{-\omega(1)}$, at least $n^{\omega(1)}$ iterations to find the global optimum of GNS. $\square$

Hence, we have completed the picture by providing fitness landscape characteristics which allow Metropolis to find the global optimum efficiently, while MAHH$_{\text{OI}}$ requires exponential time for any parameter setting.

8. Conclusions

We have presented an analysis of the performance of the Move Acceptance Hyper-Heuristic (MAHH$_{\text{OI}}$) for multimodal optimisation. The hyper-heuristic chooses at random the OnlyImproving (OI) acceptance operator with probability $1 - p$, and the AllMoves (AM) acceptance operator with probability p. All the statements given in the paper for MAHH$_{\text{OI}}$ also hold for MAHH$_{\text{IE}}$, a variant of MAHH$_{\text{OI}}$ which chooses the ImprovingAndEqual acceptance operator with probability $1 - p$, and the AllMoves acceptance operator with probability p. We summarise our findings in Table 1.

We first identified parameter values that allow the algorithm to hillclimb up to local or global optima. We have shown that setting the parameter value to $p = O((\log\log n)^{1-\varepsilon}/n)$, for any constant $\varepsilon > 0$, allows to hillclimb the OneMax function

Table 1
Comparative performance of MAHH$_{OI}$, Metropolis and the $(1+1)$ EA for the functions considered in this paper. For further details, see the relevant theorems. * Formal theorems are not presented but proofs are straightforward.

Function	MAHH$_{OI}$	Metropolis	$(1+1)$ EA
OneMax	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
Cliff$_d$	$O\left(n \log n + \frac{n^3}{d^2}\right)$	$\left(\frac{n}{\log n}\right)^{d-1/2}$	$\Theta(n^d)$
Jump$_m$	$O\left(n \log n + \frac{(1+\varepsilon)^{m-1} n^{2m}}{m^2 m!}\right)$	$2^{\Omega(n)}$	$\Theta(n^m)$
CliffJump$_{d,r,s}$ $(0 < s < 1)$	slower	faster	$\Theta(n^d)$*
CliffJump$_{d,r,s}$ $(1 < s)$	faster	slower	$\Theta(n^d)$*
GNS	$n^{\omega(1)}$	$O(n^3)$	$\Theta(n^n)$*

in expected $O(n \log n)$ runtime as desired. On the other hand, for too large parameter values we have shown how the runtime becomes exponential for any function with unique optimum.

Afterwards we considered several multimodal function classes with different characteristics which may be encountered in applications involving an optimisation process. Firstly we showed that for local optima from which a gradient of increasing fitness is close, MAHH$_{OI}$ is efficient. For the purpose we used the Cliff$_d$ function class where MAHH$_{OI}$ matches the best possible asymptotic expected runtime of any unary unbiased randomised search heuristic for any problem with up to a polynomial number of optima [3]. On the other hand, elitist search heuristics and Metropolis require exponential runtime in the distance between the local and global optima.

We then considered fitness landscapes with harder to escape local optima with large basins of attraction of decreasing fitness that have to be traversed for the global optimum to be identified. For the purpose, we considered the standard Jump$_m$ function class. While Metropolis requires exponential runtime in the problem size with overwhelming probability independent from the size of the basin of attraction of the local optima, MAHH$_{OI}$ has polynomial expected runtime for not too large basins as its runtime is only exponential in the size of the basin of attraction with overwhelming probability. We pointed out that using higher mutation rates has been shown to be beneficial compared to non-elitism in such settings. Obviously such advantages can also benefit MAHH$_{OI}$ if it is incorporated with such a mutation operator rather than just allowing it to flip one bit per iteration.

The reason for the better performance of MAHH$_{OI}$ over Metropolis in the considered landscapes is the particular sensitivity of the latter algorithm to the magnitude of the fitness difference between the parent and offspring solutions. To shed further light on this aspect we designed a more general function class called CliffJump$_{d,r,s}$ where the steepness and smoothness of the slopes of the local optima's basin of attraction may be tuned. The analysis for this function class reveals that while Metropolis may be tuned to outperform MAHH$_{OI}$ on smooth basins with gently decreasing fitness, the opposite occurs on ragged basins where fitness values may change drastically. In particular, depending on the magnitude of the negative slope of CliffJump$_{d,r,s}$, the parameters of Metropolis or MAHH$_{OI}$ can be set so as to outperform the other algorithm regardless of its choice of parameters. However, if the basin is not too large MAHH$_{OI}$ always optimises the function in polynomial time independent from the steepness of the gradient, while Metropolis also requires that the basin decreases gently to be efficient.

We conclude the paper by capitalising on this sensitivity of Metropolis to present a scenario where Metropolis significantly outperforms MAHH$_{OI}$. The GNS function, introduced originally by Jansen and Wegener [38], consists of a steep OneMax style fitness gradient followed by a path of slowly decreasing fitness surrounded by points of drastically lower fitness. While Metropolis is able to follow the gradient by ignoring the drastically worse solutions, MAHH$_{OI}$ falls off the path with overwhelming probability by accepting the points of considerably lower fitness.

Overall, apart from this pathological scenario, we believe the analysis presented in this paper provides considerable theoretical evidence of the advantages of MAHH$_{OI}$ over Metropolis for multimodal optimisation. Furthermore, our analysis of CliffJump$_{d,r,s}$ sheds light on a wide range of basins of attraction where the non-elitist MAHH$_{OI}$ outperforms elitist evolutionary algorithms even up to polynomial versus exponential runtimes. We believe that by allowing arbitrarily large radii of attraction, the benchmark function provides a much more convincing example of the importance of non-elitism for combinatorial optimisation than the commonly used Cliff$_d$ function class. Indeed, results "giving a mildly general advice which of the existing approaches to cope with local optima are preferable in which situations" have recently been deemed by Doerr as "highly desirable" [20]. We believe our contribution sheds much light on the topic.

The analysis points to various aspects that should be investigated to build upon the results. On one hand, experimental and theoretical comparisons of the two non-elitist heuristics for classical combinatorial optimisation problems should be undertaken to validate the insights presented herein or shed further light on the differences in behaviour and performance of the two forms of non-elitism. On the other hand, studies should be undertaken regarding equipping MAHH$_{OI}$ with more sophisticated mutation operators which would give the hyper-heuristic further options for escaping from local optima. Given that MAHH$_{OI}$ and Metropolis can outperform each other in different scenarios, the analysis of a HH that selects which of the two algorithms should be applied at different stages of the optimisation process may be a promising route to be explored for more general effectiveness in multimodal optimisation.

Concerning the general fields of automated algorithm selection and hyper-heuristics the use of more sophisticated move acceptance operators should be explored. One interesting option would be to allow the MAHH to automatically adapt the value of the parameter p during the optimisation process rather than keeping it static, similarly to the adaptation of the learning period for heuristic selection hyper-heuristics [25,47]. Another natural direction is to investigate when even more sophisticated move-acceptance approaches commonly used in the hyper-heuristics literature, such as *Monte Carlo* approaches [2], would lead to improved performance. Finally, more comprehensive hyper-heuristics that choose between multiple parameter sets (e.g., mutation, selection and population size) should be analysed.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgements

This work was supported by the EPSRC under Grant No. EP/M004252/1. The authors would like to thank Dirk Sudholt for suggesting how to improve the upper bound of Theorem 13.

References

[1] F. Alanazi, P.K. Lehre, Runtime analysis of selection hyper-heuristics with classical learning mechanisms, in: Proceedings of the IEEE Congress on Evolutionary Computation, CEC '14, IEEE, 2014, pp. 2515–2523.
[2] M. Ayob, G. Kendall, A Monte Carlo hyper-heuristic to optimise component placement sequencing for multi head placement machine, in: Proceedings of the International Conference on Intelligent Technologies, InTech '03, Springer, 2003, pp. 132–141.
[3] G. Badkobeh, P.K. Lehre, D. Sudholt, Black-box complexity of parallel search with distributed populations, in: Proceedings of the Workshop on Foundations of Genetic Algorithms, FOGA '15, ACM, 2015, pp. 3–15.
[4] B. Bilgin, E. Özcan, E.E. Korkmaz, An experimental study on hyper-heuristics and exam timetabling, in: Practice and Theory of Automated Timetabling, PATAT '07, Springer, 2007, pp. 394–412.
[5] E. Burke, G. Kendall, E. Soubeiga, A tabu-search hyperheuristic for timetabling and rostering, J. Heuristics 9 (6) (2003) 451–470.
[6] E.K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, R. Qu, Hyper-heuristics: a survey of the state of the art, J. Oper. Res. Soc. (2013) 1695–1724.
[7] E.K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, J.R. Woodward, A classification of hyper-heuristic approaches, in: Handbook of Metaheuristics, Springer, 2010, pp. 449–468.
[8] D. Corus, P.S. Oliveto, D. Yazdani, On the runtime analysis of the opt-IA artificial immune system, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '17, ACM, 2017, pp. 83–90.
[9] D. Corus, P.S. Oliveto, D. Yazdani, Fast artificial immune systems, in: Parallel Problem Solving from Nature, PPSN '18, Springer, 2018, pp. 67–78.
[10] D. Corus, P.S. Oliveto, D. Yazdani, Artificial immune systems can find arbitrarily good approximations for the NP-hard number partitioning problem, Artif. Intell. 274 (2019) 180–196.
[11] D. Corus, P.S. Oliveto, D. Yazdani, On inversely proportional hypermutations with mutation potential, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '19, ACM, 2019, pp. 215–223.
[12] D. Corus, P.S. Oliveto, D. Yazdani, When hypermutations and ageing enable artificial immune systems to outperform evolutionary algorithms, Theor. Comput. Sci. 832 (2020) 166–185.
[13] D. Corus, P.S. Oliveto, D. Yazdani, Automatic adaptation of hypermutation rates for multimodal optimisation, in: Proceedings of the Workshop on Foundations of Genetic Algorithms, FOGA '21, ACM, 2021, pp. 1–12.
[14] D. Corus, P.S. Oliveto, D. Yazdani, Fast immune system-inspired hypermutation operators for combinatorial optimization, IEEE Trans. Evol. Comput. 25 (5) (2021) 956–970.
[15] P. Cowling, G. Kendall, E. Soubeiga, A hyperheuristic approach to scheduling a sales summit, in: Practice and Theory of Automated Timetabling, PATAT '01, Springer, 2001, pp. 176–190.
[16] P. Cowling, G. Kendall, E. Soubeiga, Hyperheuristics: a tool for rapid prototyping in scheduling and optimisation, in: Proceedings of the Workshop on Applications of Evolutionary Computing, EvoWorkshops '02, Springer, 2002, pp. 1–10.
[17] D. Dang, T. Friedrich, T. Kötzing, M.S. Krejca, P.K. Lehre, P.S. Oliveto, D. Sudholt, A.M. Sutton, Escaping local optima using crossover with emergent diversity, IEEE Trans. Evol. Comput. 22 (3) (2018) 484–497.
[18] D.-C. Dang, A. Eremeev, P.K. Lehre, Escaping local optima with non-elitist evolutionary algorithms, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, 2021, pp. 12275–12283.
[19] D.-C. Dang, T. Friedrich, T. Kötzing, M.S. Krejca, P.K. Lehre, P.S. Oliveto, D. Sudholt, A.M. Sutton, Escaping local optima with diversity mechanisms and crossover, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '16, ACM, 2016, pp. 645–652.
[20] B. Doerr, Does comma selection help to cope with local optima?, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '20, ACM, 2020, pp. 1304–1313.
[21] B. Doerr, C. Doerr, Theory of parameter control for discrete black-box optimization: provable performance gains through dynamic parameter choices, in: B. Doerr, F. Neumann (Eds.), Theory of Evolutionary Computation - Recent Developments in Discrete Optimization, Springer, 2020, pp. 271–321.
[22] B. Doerr, C. Doerr, J. Yang, k-bit mutation with self-adjusting k outperforms standard bit mutation, in: Proceedings of the International Conference on Parallel Problem Solving from Nature, PPSN '16, Springer, 2016, pp. 824–834.
[23] B. Doerr, D. Johannsen, C. Winzen, Drift analysis and linear functions revisited, in: IEEE Congress on Evolutionary Computation, CEC '10, IEEE, 2010, pp. 1–8.
[24] B. Doerr, H.P. Le, R. Makhmara, T.D. Nguyen, Fast genetic algorithms, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '17, ACM, 2017, pp. 777–784.
[25] B. Doerr, A. Lissovoi, P.S. Oliveto, J.A. Warwicker, On the runtime analysis of selection hyper-heuristics with adaptive learning periods, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '18, ACM, 2018, pp. 1015–1022.

[26] B. Doerr, D. Sudholt, C. Witt, When do evolutionary algorithms optimize separable functions in parallel?, in: Proceedings of the Workshop on Foundations of Genetic Algorithms, FOGA '13, ACM, 2013, pp. 51–64.
[27] J.H. Drake, A. Kheiri, E. Özcan, E.K. Burke, Recent advances in selection hyper-heuristics, Eur. J. Oper. Res. 285 (2) (2020) 405–428.
[28] S. Droste, Analysis of the $(1+1)$ EA for a noisy OneMax, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '04, Springer, 2004, pp. 1088–1099.
[29] S. Droste, T. Jansen, I. Wegener, Dynamic parameter control in simple evolutionary algorithms, in: Proceedings of the Workshop on Foundations of Genetic Algorithms, FOGA '01, ACM, 2001, pp. 275–294.
[30] S. Droste, T. Jansen, I. Wegener, On the analysis of the $(1+1)$ evolutionary algorithm, Theor. Comput. Sci. (2002) 51–81.
[31] T. Friedrich, F. Quinzan, M. Wagner, Escaping large deceptive basins of attraction with heavy-tailed mutation operators, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '18, ACM, 2018, pp. 293–300.
[32] G.T. Hall, P.S. Oliveto, D. Sudholt, On the impact of the performance metric on efficient algorithm configuration, Artif. Intell. 303 (2022) 103629.
[33] V. Hasenöhrl, A.M. Sutton, On the runtime dynamics of the compact genetic algorithm on jump functions, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '18, ACM, 2018, pp. 967–974.
[34] J. He, X. Yao, Drift analysis and average time complexity of evolutionary algorithms, Artif. Intell. 127 (1) (2001) 57–85.
[35] M.A. Hevia Fajardo, D. Sudholt, Self-adjusting offspring population sizes outperform fixed parameters on the cliff function, in: Proceedings of the Workshop on Foundations of Genetic Algorithms, FOGA '21, ACM, 2021, pp. 1–15.
[36] J. Jägersküpper, T. Storch, When the plus strategy outperforms the comma strategy and when not, in: Proceedings of IEEE Symposium on Foundations of Computational Intelligence, FOCI '07, IEEE, 2007, pp. 25–32.
[37] T. Jansen, Simulated annealing, in: A. Auger, B. Doerr (Eds.), Theory of Randomized Search Heuristics, World Scientific, 2011, pp. 171–195.
[38] T. Jansen, I. Wegener, A comparison of simulated annealing with a simple evolutionary algorithm on pseudo-boolean functions of unitation, Theor. Comput. Sci. (2007) 73–93.
[39] A. Juels, M. Wattenberg, Stochastic hillclimbing as a baseline method for evaluating genetic algorithms, in: Proceedings of the 8th International Conference on Neural Information Processing Systems, NIPS '95, MIT Press, 1995, pp. 430–436.
[40] L. Kotthoff, Algorithm selection for combinatorial search problems: a survey, AI Mag. 35 (3) (2014) 48–60.
[41] T. Kötzing, D. Sudholt, M. Theile, How crossover helps in pseudo-boolean optimization, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '11, ACM, 2011, pp. 989–996.
[42] P.K. Lehre, P.S. Oliveto, Theoretical analysis of stochastic search algorithms, in: M.G.C. Resende, R. Marti, P.M. Pardalos (Eds.), Handbook of Heuristics, Springer, 2018, pp. 1–36.
[43] P.K. Lehre, E. Özcan, A runtime analysis of simple hyper-heuristics: to mix or not to mix operators, in: Proceedings of the Workshop on Foundations of Genetic Algorithms, FOGA '13, ACM, 2013, pp. 97–104.
[44] P.K. Lehre, C. Witt, Black-box search by unbiased variation, Algorithmica (2012) 623–642.
[45] J. Lengler, Drift analysis, in: B. Doerr, F. Neumann (Eds.), Theory of Evolutionary Computation: Recent Developments in Discrete Optimization, Springer, 2020, pp. 89–131.
[46] A. Lissovoi, P.S. Oliveto, J.A. Warwicker, On the time complexity of algorithm selection hyper-heuristics for multimodal optimisation, in: Proceedings of the AAAI Conference on Artificial Intelligence, AAAI '19, 2019, pp. 2322–2329.
[47] A. Lissovoi, P.S. Oliveto, J.A. Warwicker, How the duration of the learning period affects the performance of random gradient selection hyper-heuristics, in: Proceedings of the AAAI Conference on Artificial Intelligence, AAAI '20, 2020, pp. 2376–2383.
[48] A. Lissovoi, P.S. Oliveto, J.A. Warwicker, Simple hyper-heuristics control the neighbourhood size of randomised local search optimally for LeadingOnes, Evol. Comput. 28 (3) (2020) 437–461.
[49] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller, Equation of state calculations by fast computing machines, J. Chem. Phys. (1953) 1087–1092.
[50] A. Nareyek, Choosing search heuristics by non-stationary reinforcement learning, in: Metaheuristics: Computer Decision-Making, Springer, 2004, pp. 523–544.
[51] P.S. Oliveto, Rigorous performance analysis of hyper-heuristics, in: N. Pillay, R. Qu (Eds.), Automated Design of Machine Learning and Search Algorithms, Springer, 2021, pp. 45–71.
[52] P.S. Oliveto, C. Witt, Simplified drift analysis for proving lower bounds in evolutionary computation, Algorithmica (2011) 369–386.
[53] P.S. Oliveto, C. Witt, Erratum: Simplified drift analysis for proving lower bounds in evolutionary computation, CoRR, arXiv:1211.7184 [abs], 2012.
[54] P.S. Oliveto, C. Witt, Improved time complexity analysis of the simple genetic algorithm, Theor. Comput. Sci. 605 (2015) 21–41.
[55] E. Özcan, B. Bilgin, E.E. Korkmaz, Hill climbers and mutational heuristics in hyperheuristics, in: Parallel Problem Solving from Nature, PPSN '06, Springer, 2006, pp. 202–211.
[56] E. Özcan, B. Bilgin, E.E. Korkmaz, A comprehensive analysis of hyper-heuristics, Intell. Data Anal. 12 (1) (2008) 3–23.
[57] T. Paixão, J. Pérez Heredia, D. Sudholt, B. Trubenová, Towards a runtime comparison of natural and artificial evolution, Algorithmica (2017) 681–713.
[58] N. Pillay, R. Qu, Hyper-Heuristics: Theory and Applications, Springer, 2018.
[59] M. Preuss, Multimodal Optimization by Means of Evolutionary Algorithms, Natural Computing Series, Springer, 2015.
[60] C. Qian, K. Tang, Z.-H. Zhou, Selection hyper-heuristics can provably be helpful in evolutionary multi-objective optimization, in: Parallel Problem Solving from Nature, PPSN '16, Springer, 2016, pp. 835–846.
[61] A. Rajabi, C. Witt, Self-adjusting evolutionary algorithms for multimodal optimization, in: Proceedings of the 2020 Genetic and Evolutionary Computation Conference, GECCO '20, ACM, 2020, pp. 1314–1322.
[62] A. Rajabi, C. Witt, Self-adjusting evolutionary algorithms for multimodal optimization, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '21, ACM, 2021, pp. 1178–1186.
[63] A. Rajabi, C. Witt, Stagnation detection with randomized local search, in: European Conference on Evolutionary Computation in Combinatorial Optimization, EvoCOP '21, 2021, pp. 152–168.
[64] T. Stützle, M. López-Ibáñez, Automated design of metaheuristic algorithms, in: M. Gendreau, J.-Y. Potvin (Eds.), Handbook of Metaheuristics, Springer, 2019, pp. 541–579.
[65] D. Sudholt, A new method for lower bounds on the running time of evolutionary algorithms, IEEE Trans. Evol. Comput. 17 (3) (2013) 418–435.