On the Time Complexity of Algorithm Selection Hyper-Heuristics for Multimodal Optimisation

Andrei Lissovoi and Pietro S. Oliveto and John Alasdair Warwicker
Rigorous Research, Department of Computer Science
The University of Sheffield, Sheffield S1 4DP, United Kingdom
{a.lissovoi,p.oliveto,j.warwicker}@sheffield.ac.uk

Abstract

Selection hyper-heuristics are automated algorithm selection methodologies that choose between different heuristics during the optimisation process. Recently selection hyper-heuristics choosing between a collection of elitist randomised local search heuristics with different neighbourhood sizes have been shown to optimise a standard unimodal benchmark function from evolutionary computation in the optimal expected runtime achievable with the available low-level heuristics. In this paper we extend our understanding to the domain of multimodal optimisation by considering a hyper-heuristic from the literature that can switch between elitist and non-elitist heuristics during the run. We first identify the range of parameters that allow the hyper-heuristic to hillclimb efficiently and prove that it can optimise a standard hillclimbing benchmark function 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 the hyper-heuristic is efficient by swiftly escaping local optima and ones where it is not. For a function class called CLIFF, where a new gradient of increasing fitness can be identified after escaping local optima, the hyper-heuristic is extremely efficient while a wide range of established elitist and non-elitist algorithms are not, including the well-studied Metropolis algorithm. We complete the picture with an analysis of another standard benchmark function called JUMP as an example to highlight problem characteristics where the hyper-heuristic is inefficient. Yet, it still outperforms the well-established non-elitist Metropolis algorithm.

1 Introduction

Selection hyper-heuristics 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 (Cowling, Kendall, and Soubeiga 2001). Rather than deciding in advance which heuristics to apply for a problem, the aim 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 (Cowling, Kendall, and Soubeiga 2001; 2002), they have since been successfully applied to a variety of hard combinatorial optimisation problems (see Burke et al. 2010; 2013) for surveys of results.

Despite their numerous successes, very limited rigorous theoretical understanding of their behaviour and performance is available (Lehre and Özcan 2013; Alanzai and Lehre 2014). Recently, it has been proved that a simple selection hyper-heuristic called Random Descent (Cowling, Kendall, and Soubeiga 2001; 2002), choosing between elitist randomised local search (RLS) heuristics with different neighbourhood sizes, optimises a standard unimodal benchmark function from evolutionary computation called LEADINGONES in the best possible runtime (up to lower order terms) achievable with the available low-level heuristics (Lissovoi, Oliveto, and Warwicker 2018). For this to occur, it is necessary to run the selected low-level heuristics for a sufficient amount of time, called the learning period, to allow the hyper-heuristic to accurately determine how useful the chosen operator is at the current stage of the optimisation process. If the learning period is not sufficiently long, the hyper-heuristic fails to identify whether the chosen low-level heuristics are actually useful, leading to random choices of which heuristics to apply and a degradation of the overall performance. Since the optimal duration of the learning period may change during the optimisation, (Doerr et al. 2018) have recently introduced a self-adjusting mechanism and rigorously proved that it allows the hyper-heuristic to track the optimal learning period throughout the optimisation process for LEADINGONES.

In this paper we aim to extend the understanding of the behaviour and performance of hyper-heuristics to multimodal optimisation problems. In order to evaluate their capability at escaping local optima, we consider elitist and non-elitist selection operators that have been used in hyper-heuristics in the literature. (Cowling, Kendall, and Soubeiga 2001; 2002) introduced two variants of 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. Another selection operator that has been considered in the literature is the IMPROVINGANDEQUAL (IE) acceptance operator which in addition to accepting improving solutions also accepts solutions of equal quality (Ayob and Kendall 2003; Bilgin, Özcan, and Korkmaz 2007; Özcan, Bilgin, and Kork-
heuristic is asymptotically as efficient as the best mutation-that for any to locate local optima efficiently. In particular, we prove p values for tors with probability 1 s presented a function where it is necessary to mix the two ac-
rection, for instance to escape from local optima.

(Qian, Tang, and Zhou 2016) analysed selection hyper-
heuristics in the context of multi-objective optimisation. They considered a hyper-heuristic selecting between elitist (IE) and strict elitist (OI) acceptance operators and pre-
nted a function where it is necessary to mix the two ac-
ance operators. Lehre and Özcan presented a hyper-
heuristic which chooses between two different low level heuristics that use the above described OI (elitist) and AM (non-elitist) acceptance operators (Lehre and Özcan 2013). The considered Simple Random hyper-heuristic uses 1-bit flips as 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 Random Local Search algorithm that switches between strict elitism, by only accepting improvements, and extreme non-elitism by accepting any new found solution. For the standard ROYALROAD benchmark function from evolutionary computation, which consists of several blocks of k bits each that have to be set correctly to observe a fitness improve-
ment, and k ≥ 2 they theoretically proved that it is necessary to mix the acceptance operators for the hyper-heuristic to be efficient, because by only accepting improvements (i.e., p = 0) the runtime is infinite due to the hyper-heuristic not being able to cross the plateaus of equal fitness while by al-
ways 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 hyper-heuristic of O(n^3, k^{2k−3}) versus the O(n log(n) · (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 as good as that of their parent (Doerr, Sudholt, and Witt 2013). 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 hyper-heuristic considered in (Lehre and Özcan 2013) for multimodal optimisation problems where the considerable advantages of changing the selection operator during the run may be highlighted. In particular, we will increase our understanding of the behaviour and performance of hyper-
heuristics by providing examples of instance classes where the hyper-heuristic is efficient at escaping local optima and examples where it is 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 the hyper-heuristic to hillclimb, hence to locate local optima efficiently. In particular, we prove that for any p = 1/(1+ɛ)n, for any constant ɛ > 0, the hyper-
heuristic is asymptotically as efficient as the best mutation-
based unbiased randomised search heuristic (Lehre and Witt 2012) even though it does not rely on elitism. Afterwards we highlight the power of the hyper-heuristic 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 CLIFFJ class of functions, consisting of a local optimum that, once escaped, allows the identification of a new slope of increasing fitness. For this class of functions we rigorously prove that the hyper-
heuristic efficiently escapes the local optima and even achieves the best known expected runtime of O(n log n) (for general purpose randomised search heuristics) on the hardest instances of the function class (Corus, Oliveto, and Yazdani 2017). Thus, we prove that it considerably outperforms estabilished elitist and non-elitist evolutionary algorithms and the popular METROPOLIS algorithm. We complete the picture by considering the standard JUMPn multimodal instance class of functions to provide an example of problem characteristics where the hyper-heuristic is not efficient. This class is chosen to isolate problems where escaping local optima is very hard because it is difficult to identify a new slope of increasing gradient. Nevertheless, the hyper-heuristic is efficient for instances of moderate jump size (i.e., constant) where it still considerably outperforms METROPOLIS.

Due to space constraints in this extended abstract, we omit some proofs.

## 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.

### Algorithms

We will analyse the Move Acceptance hyper-heuristic previously considered in (Lehre and Özcan 2013). 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 IE acceptance operator is used instead of the OI operator. Thus, in MAHH_IE, the AM acceptance operator

---

**Algorithm 1** Move Acceptance Hyper-Heuristic (MAHH_{OI}) (Lehre and Özcan 2013)

1: Choose $x \in \{0, 1\}^n$ uniformly at random
2: while termination criteria not satisfied do
3: $x' \leftarrow \text{FLIPRANDOMBIT}(x)$
4: $\text{ACC} \leftarrow \begin{cases} \text{ALLMOVES} & \text{with probability } p \\ \text{ONLYIMPROVING} & \text{otherwise} \end{cases}$
5: if ACC($x, x'$) then $x \leftarrow x'$
is used with probability $p$, and the IE acceptance operator is used with probability $1 - p$.

The problem classes we consider are all functions of unitation, where 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. Under these conditions, we point out that all the statements made in the paper for the MAHH$_{O1}$ hyper-heuristic will also hold for the MAHH$_{IE}$ hyper-heuristic. In the rest of this paper, we will focus the analysis on the MAHH$_{O1}$ hyper-heuristic.

**Benchmark Function Classes**

We will present a runtime analysis of the MAHH$_{O1}$ hyper-heuristic on three benchmark problem classes defined over $n$-bit strings – ONE MAX, CLIFF$_d$, JUMP$_m$ – 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 ONE MAX 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 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:

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

In the above definition the global optimum is placed in the 1$^n$ bit-string for convenience of the analysis i.e., 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 Hamming distance to the global optimum. This class of functions is known to be the easiest among all functions with a unique global optimum for unary unbiased black box algorithms (mutation-based EAs) (Lehre and Witt 2012).

We also consider two similar multi-modal problem classes, where the optimisation algorithm will need to escape from a local optimum in order to reach the global optimum. The two classes differ in whether the fitness function guides the search toward or away from the global optimum once the search process does escape the local optimum.

The CLIFF$_d$ class of functions was originally proposed as an example where non-elitist evolutionary algorithms outperform elitist ones (Jägersküpper and Storch 2007). 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 Figure 1. We define the CLIFF$_d$ class of functions (for $1 < d < n/2$) as follows:

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

As for ONE MAX, 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 toward the local optima: thus, to find the global optimum, the search algorithm 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 Figure 2. We define the JUMP$_m$ class of functions (for $1 < m < n/2$) as follows:

$$\text{JUMP}_m(x) := \begin{cases} n + m & \text{if } |x|_1 = n, \\ m + \text{ONE MAX}(x) & \text{if } |x|_1 \leq n - m, \\ n - \text{ONE MAX}(x) & \text{otherwise}. \end{cases}$$

As for ONE MAX, the global optimum is placed at the 1$^n$ bit string to simplify notation. Compared to 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.

**Mathematical Analysis Tools**

We now present some well known drift analysis theorems often used to bound the expected runtime of a randomised
search heuristic by considering its drift (i.e., its expected decrease in distance from the optimum in each step). We will apply these 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 (He and Yao 2001)). Let $\{X_t\}_{t \geq 0}$ 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 $\delta_a \geq \delta_b > 0$ such that for all $t \geq 0$, we have

$$\delta_a \geq E(X_t - X_{t+1} | X_t) \geq \delta_b,$$

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

$$X_0 / \delta_a \leq E(T | X_0) \leq X_0 / \delta_b,$$

and

$$E(X_0) / \delta_a \leq E(T) \leq E(X_0) / \delta_b.$$

If the drift changes considerably throughout the process, then the following Variable Drift Theorem provides sharper upper bounds on the runtime.

**Theorem 2** (Variable Drift Theorem (Johannsen 2010)). Let $\{X_t\}_{t \geq 0}$ be a stochastic process over some state space $S \subseteq \{0\} \cup [x_{\min}, x_{\max}]$, where $x_{\min} > 0$. Let $h(x)$ be an integrable, monotone increasing function on $[x_{\min}, x_{\max}]$ such that $E(X_t - X_{t+1} | X_t \geq x_{\min}) \geq h(X_t)$. Then it holds for the first hitting time $T := \min\{t \mid X_t = 0\}$ that

$$E(T | X_0) \leq \frac{x_{\min}}{h(x_{\min})} + \int_{x_{\min}}^{X_0} \frac{1}{h(x)} dx.$$

The Negative Drift Theorem allows to derive exponential lower bounds on the runtime when the drift is negative in large enough areas of the search space (Oliveiro and Witt 2011; 2012). The Simplified Drift Theorem with Scaling is a generalised version of the Negative Drift Theorem that allows the negative drift $\varepsilon$ to be sub-constant in magnitude.

**Theorem 3** (Simplified Drift Theorem with Scaling (Oliveiro and Witt 2015)). 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. $E(X_{t+1} - X_t | X_0, \ldots, X_t; a < X_t < b) \geq \varepsilon$.
2. $P(|X_{t+1} - X_t| \geq \ell r | X_0, \ldots, X_t; a < X_t < b) \leq e^{-r}$ 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 $T^* := \min\{t \geq 0 : X_t \leq a | X_0 > b\}$ it holds that $P(T^* \leq e^{\ell t/(132r^2)}) = O(e^{-\ell t/(132r^2)})$.

3 Unimodal Functions

We begin the study of MAHH$_{OI}$ by analysing its performance on unimodal functions. Since in this setting accepting worsening moves is never required, the hyper-heuristic cannot outperform the elitist algorithms. Nevertheless, Theorem 6 shows that MAHH$_{OI}$ 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 4.** The runtime of MAHH$_{OI}$ on ONEMAX($x$), with $p \geq (\sqrt{\pi} \log^2 n)/n$, is at least $n^{\Omega(\log n)}$ with probability at least $1 - n^{-\Omega(\log n)}$.

For ONEMAX, the larger the value of $p$, the greater the probability of accepting a move away from the global optimum and the greater the expected runtime of the hyper-heuristic. Applying the standard Negative Drift Theorem (Oliveiro and Witt 2011; 2012) for MAHH$_{OI}$ with any constant value of $p$ gives the following result.

**Theorem 5.** The runtime of MAHH$_{OI}$ on ONEMAX, with $p = \Theta(1)$, is at least $2^{\Omega(n)}$ with probability at least $1 - 2^{-\Omega(n)}$.

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

**Theorem 6.** The expected runtime of MAHH$_{OI}$ on ONEMAX, with $p < 1/(n - 1)$, is at most

$$E(T_p) \leq \frac{n}{1 - p(n - 1)} + \frac{n}{1 + p} \cdot \ln \left(\frac{n}{1 - p(n - 1)}\right).$$

If $p = 0$, we have the well-studied Randomised Local Search (RLS) algorithm, and Theorem 6 gives an expected runtime of $E(T_0) \leq n + n \ln n = O(n \log n)$. For $p = 1/n$, combining Theorem 6 with a separate argument regarding the time needed to reach a fitness of $n$ from a fitness of $n - 1$ (using e.g. Lemma 8) yields a $O(n \log n)$ bound on the expected runtime. For $p = \frac{1}{1 + \epsilon/n}$ with any constant $\epsilon > 0$, applying Theorem 6 directly shows that any ONEMAX style hillclimbs can be completed in expected time $O(n \log n)$.

**Corollary 7.** The expected runtime of MAHH$_{OI}$ on ONEMAX, with $p = \frac{1}{1 + \epsilon/n}$, for any constant $\epsilon > 0$, is $O(n \log n)$.

For the remainder of the paper, we consider MAHH$_{OI}$ with $p = \frac{1}{1 + \epsilon/n}$. We will show how with this parameter value, as well as hill-climbing efficiently, the hyper-heuristics can escape from difficult local optima effectively.

4 When the Hyper-Heuristic is Efficient

We now analyse the CLIFF$_d$ ($1 < d < n/2$) class of benchmark functions. Recall that CLIFF$_d$ features a local optimum that the hyper-heuristic must escape before climbing a fitness gradient toward the global optimum. We refer to the local optimum at $i = n - d$ as the ‘cliff’, two ONEMAX style hillclimbs as the ‘first slope’ and ‘second slope’ respectively (see Figure 1), 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, by either dropping
down from the cliff and accepting a worse candidate solution and then climbing up the second slope, or 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 length of the cliff, but not with local mutations).

We now consider the performance of MAHH\textsubscript{O1} on CLIFF\textsubscript{d}. Clearly, with \( p = 1 \), MAHH\textsubscript{O1} 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\textsubscript{O1} with \( p = \frac{1}{(1+\epsilon)n} \), which has been shown to hillclimb efficiently (Corollary 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 9 bounds the expected runtime of MAHH\textsubscript{O1} on CLIFF\textsubscript{d} from above. We begin, however, by introducing a helper Lemma, which was proved in (Droste, Jansen, and Wegener 2001), for trajectory based algorithms which can only change the number of 1-bits in the bit-string by 1. The lemma was subsequently used to analyse the performance of the (1+1) EA for noisy OneMax for small noise strength (Droste 2004). 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 8. (Droste, Jansen, and Wegener 2001, Lemma 3)
Let \( E(T_1^+) \) be the expected time to reach a state with \( i + 1 \) 1-bits, given a state with \( i \) 1-bits, and \( p_1^+ \) and \( p_i^- \) be the transition probabilities to reach a state with, respectively, \( i + 1 \) and \( i - 1 \) 1-bits. Then:

\[
E(T_1^+) = \frac{1}{p_i^+} + \frac{p_i^-}{p_i^+} \cdot E(T_{i-1}^+).
\]

Within the context of non-elitist local search algorithms, such as MAHH\textsubscript{O1}, the transition probability \( p_1^+ (p_i^-) \) 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 9. The expected runtime of MAHH\textsubscript{O1} on CLIFF\textsubscript{d}, with \( p = \frac{1}{(1+\epsilon)n} \) for any constant \( \epsilon > 0 \), is \( O \left( n \log n + \frac{n^2}{\epsilon^2} \right) \).

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 \( i \geq n - d + 1 \), the second when \( i \geq n - d + 2 \), 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,

\[
E(T) = E(T_1) + E(T_2) + E(T_3) + E(T_4). \tag{1}
\]

In the first stage, while \( i < n - d \), CLIFF\textsubscript{d} resembles the OneMax function, and we can use the upper bound for the expected runtime of MAHH\textsubscript{O1} on OneMax with \( p = \frac{1}{(1+\epsilon)n} \) from Theorem 6. Hence, \( E(T_1) = O(n \log n) \).

The second stage begins with \( i \geq n - d \), and ends once \( i \geq n - d + 1 \). 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 \( \frac{1}{(1+\epsilon)n} \). 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 8 to bound \( E(T_2) \), the expected time to jump down from the cliff,

\[
E(T_2) = E(T_{d-d}^+) = \frac{n^2(1+\epsilon)}{d} + \frac{n-d}{d} \cdot E(T_{n-d-1}^+). \tag{2}
\]

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

\[
\Delta(n-d-1) = \frac{d+1}{n} - \frac{1}{(1+\epsilon)n} \cdot \frac{n-d-1}{n} = \frac{n(d+\epsilon(d+1)) + d + 1}{n^2(1+\epsilon)},
\]

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 \( p = 1/(1+\epsilon)n \).

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

\[
E(T_{n-d-1}^+) \leq 1/\Delta(n-d-1) = \frac{n^2(1+\epsilon)}{n(d+\epsilon(d+1)+1) + d + 1} = O \left( \frac{n}{d} \right).
\]

We can now return to bounding \( E(T_2) \) in Equation 2:

\[
E(T_2) = \frac{n^2(1+\epsilon)}{d} + \frac{n-d}{d} \cdot E(T_{n-d-1}^+) = \frac{n^2(1+\epsilon)}{d} + \frac{n-d}{d} \cdot O \left( \frac{n}{d} \right) = O \left( \frac{n^2}{d} \right). \tag{3}
\]

The third stage begins with \( i \geq n - d + 1 \), and ends once \( i \geq n - d + 2 \). 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
If $d = 2$, the CLIFF$_d$ function will have been optimised at this point. However, for $d \geq 3$, 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 8 bound

$$E(T_{n-d+2}) = \frac{n}{d-2} + \frac{1}{(1+\epsilon)n} \cdot \frac{n-d+2}{d-2} \cdot E(T_{n-d+1})$$

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 $E(T_4 \mid d = o(n)) \leq O(1) \cdot (n^3/d^3) = O(n^3/d^2)$. However, if $d$ is linear in the problem size, the first terms will dominate, and we will have:

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

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 1 and complete the proof:

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

Algorithm 2 METROPOLIS Algorithm (Metropolis et al. 1953)

1. Choose $x \in \{0, 1\}^n$ uniformly at random, set $\alpha(n) \geq 1$
2. while termination criteria not satisfied do
3. $x' \leftarrow \text{FLIPRANDOMBIT}(x)$
4. $\Delta f \leftarrow f(x') - f(x)$
5. if $\Delta f \geq 0$ then $x \leftarrow x'$
6. else Choose $r \in [0, 1]$ uniformly at random
7. if $r \leq \alpha(n) \frac{f(x)}{d}$ then $x \leftarrow x'$

Theorem 9 gives an expected runtime bound of MAHH$_{O1}$ on 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 an artificial immune system, which escapes the local optimum with an ageing operator (also $O(n \log n)$ in expectation (Corus, Oliveto, and Yazdani 2017)), which is the best runtime known for hard CLIFF$_d$ functions (for problem-independent search heuristics). We suspect MAHH$_{O1}$ is faster in practice (i.e., by decreasing the leading constants in the runtime), but leave this proof for future work. Mutation based EAs have a runtime of $\Theta(n^d)$ for $d \leq n/2$ (Paixão et al. 2017); for $d = \Theta(n)$, this will give at least exponential runtime in the length of the cliff. Steady-State Genetic Algorithms which use crossover have recently been proven to be faster by at least a linear factor (Dang et al. 2018), but would still require exponential expected runtimes for large cliff lengths.

The worst-case scenario for MAHH$_{O1}$ is a constant cliff length, giving a runtime of $O(n^3)$. This means that the $(1+1)$ EA will outperform MAHH$_{O1}$ if $d < 3$, but will be slower for any $3 < d \leq n/2$, with a performance gap that increases exponentially with the length of the cliff. For CLIFF$_d$, only an upper bound on the expected runtime of the non-elitist $(1,\lambda)$ EA is available in the literature: $O(n^{22})$ if $\lambda$ is neither too large nor too small (Jürgenskühper and Storch 2007).

We can also compare the performance of MAHH$_{O1}$ on CLIFF$_d$ with a well-studied non-elitist algorithm. The METROPOLIS algorithm (Algorithm 2) accepts worsening moves with probability $\alpha(n) f(x')/(d + f(x))$, for some $\alpha(n) \geq 1$, and accepts all improvements (Metropolis et al. 1953). Theorem 10 shows that MAHH$_{O1}$ will considerably outperform METROPOLIS on CLIFF$_d$ for all $d > 3$.

Theorem 10. The expected runtime of METROPOLIS on CLIFF$_d$ is at least min $\left\{ \frac{1}{2} \cdot \frac{n^2}{d^2}, \frac{n^3/2}{d^2}, \frac{n^3}{d^2} \right\}$.

We have proven that while METROPOLIS cannot optimise hard CLIFF$_d$ variants in polynomial time, MAHH$_{O1}$ is extremely efficient for hard cliffs, and has an expected runtime of $O(n^3)$ in the worst case.
5 When the Hyper-Heuristic is Inefficient

We now consider the JUMP\textsubscript{m} function class (1 < m < n/2) as an example of a multimodal function where MAHH\textsubscript{OI} has a harder time escaping the local optimum. Unlike CLIFF\textsubscript{d}, the fitness decreases on the second slope, making it harder to traverse.

In order to optimise the JUMP\textsubscript{m} function (Figure 2), it is first necessary to reach 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 11.** The runtime of MAHH\textsubscript{OI} on JUMP\textsubscript{m}, with \( p = \frac{1}{(1 + \epsilon)m} \), is at least \( \Omega(n \log n + 2^m) \), for some constant \( \epsilon > 0 \) and any constant \( \epsilon > 0 \), with probability at least \( 1 - 2^{-\Omega(m)} \).

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

**Theorem 12.** The expected runtime of MAHH\textsubscript{OI} on JUMP\textsubscript{m}, with \( p = \frac{1}{(1 + \epsilon)m} \), for any constant \( \epsilon > 0 \), is \( O(n \log n + n^{2^{m-1}}/m) \).

Using global mutations to jump can achieve faster expected runtimes. The expected runtime of the (1+1) EA on JUMP\textsubscript{m} is \( \Theta(n^m) \) (Droste, Jansen, and Wegener 2002), and a bound on Steady State (1+1) Genetic Algorithms of \( O(n^{m-1}) \) has recently been proved (Dang et al. 2018), both outperforming our upper bound for MAHH\textsubscript{OI}. Recent work has shown that exponential speedups are achieved by an Artificial Immune System and an Evolutionary Algorithm and Genetic Algorithm (GA) with heavy-tailed mutation operators (Corus, Oliveto, and Yazdani 2018; Friedrich, Quinzan, and Wagner 2018; Doerr et al. 2017) (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 \( m = o(n) \) (Hasenöhrl and Sutton 2018). In particular for logarithmic jump lengths the algorithm optimises JUMP\textsubscript{m} in expected polynomial time. We point out that steady state GASs with diversity mechanisms that enhance the power of crossover can optimise JUMP\textsubscript{m} in expected time \( O(m n \log n + 4^m) \) for jumps up to \( m = n/8 \) (Dang et al. 2016) and in \( \Theta(n \log n + 4^m) \) with an unrealistically small crossover probability of \( O(n/m) \) (Kötzing, Sudholt, and Theile 2011).

Considering algorithms that use non-elitism to escape from local optima, we will now show that METROPOLIS has worse performance than MAHH\textsubscript{OI} on JUMP\textsubscript{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 \( a(n) \) is set high enough to enable efficient hill-climbing of the first slope.

**Theorem 13.** The runtime of METROPOLIS on JUMP\textsubscript{m}(x) is \( 2^{O(n)} \), with probability at least \( 1 - 2^{-\Omega(n)} \).

Unlike METROPOLIS, MAHH\textsubscript{OI} does not consider the magnitude of the fitness difference between two solutions in its acceptance operators. This allows it to optimise JUMP\textsubscript{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 Conclusions

We have rigorously analysed the performance of the Move Acceptance Hyper-Heuristic (MAHH\textsubscript{OI}) for multimodal optimisation problems. We have shown that setting the parameter value to \( p \leq 1/(1 + \epsilon)n \), for any constant \( \epsilon > 0 \), allows to hillclimb the ONEMAX function 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.

We have also shown that MAHH\textsubscript{OI} performs well on multimodal landscapes where the candidate solution identifies a gradient of increasing fitness after leaving a local optimum. The CLIFF\textsubscript{d} function class was analysed to provide such an example where MAHH\textsubscript{OI} exhibits very good performance, matching the best runtime known for problem-independent search heuristics, for instances that are prohibitive for elitist Evolutionary Algorithms and METROPOLIS.

On multimodal functions with long slopes of decreasing fitness that have to be traversed, MAHH\textsubscript{OI} does not perform as well. In particular, we analysed the JUMP\textsubscript{m} function class to provide such an example, proving that MAHH\textsubscript{OI} has a runtime that is exponential in the size of the ‘jump’, with overwhelming probability. However, also the established non-elitist METROPOLIS algorithm was shown to be inefficient on this benchmark function, with an even worse runtime (i.e., at least exponential in the problem size, with overwhelming probability). The difference is particularly dramatic for small jumps (i.e., \( m = \Theta(1) \)), where MAHH\textsubscript{OI} is able to find the global optimum in expected polynomial time, because unlike METROPOLIS, its acceptance operators do not consider the magnitude of the fitness difference between the parent and offspring solutions.

All the statements given in the paper for MAHH\textsubscript{OI} also hold for MAHH\textsubscript{B}, a variant of MAHH\textsubscript{OI} which chooses the IMPROVINGANDEQUAL acceptance operator with probability \( 1 - p \), and the ALLMOVES acceptance operator with probability \( p \).

Various avenues can be explored in future work. Firstly, analysing more sophisticated move acceptance operators would give more insight into the behaviour and performance of hyper-heuristics commonly used in literature, such as Monte Carlo approaches (Ayob and Kendall 2003). Also, implementing learning mechanisms within the MAHH\textsubscript{OI}, such that it could ‘learn’ to prefer a certain operator in certain parts of the fitness landscape similarly to the hyper-heuristic introduced in (Doerr et al. 2018) for mutation operator selection, should be considered. Furthermore, MAHH\textsubscript{OI} should be analysed on classical problems from combinatorial optimisation with real-world applications. Finally, more comprehensive hyper-heuristics that choose between multiple parameter sets, including the population size, should be analysed.
Acknowledgements This work was supported by the EP-SRC under Grant No. EP/M004252/1.

References


