On the Effectiveness of Random Gradient Hyper-heuristics for Multimodal Optimisation

Yuxuan Ma[a], Pietro S. Oliveto[a], John Alasdair Warwicker[b]

[a]*Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China*
[b]*Institute of Operations Research, Karlsruhe Insitute of Technology, Karlsruhe, Germany*

Abstract

Selection hyper-heuristics (SHHs) select from a set of low-level heuristics which to apply during the optimisation process. One such approach, namely the random gradient SHH, which continues to apply a randomly selected heuristic as long as it remains successful, has been shown to be able to effectively select heuristics leading to optimal expected runtimes on a range of unimodal functions. In this work, we extend the analysis of the random gradient SHH to multimodal optimisation problems to assess their performance at escaping from local optima. We consider the TwoRates benchmark function which includes several consecutive local optima separated by gaps of two alternating different sizes. The function was recently introduced to assess the performance of the *flex-EA* that uses an archive to store and re-apply the two most suitable Randomized Local Search (RLS$_k$) operators to make the jumps of different lengths. We show that the SHH can optimise the function considerably faster by identifying and consecutively re-applying the single best heuristic to overcome all of the local optima. This performance also holds when the set of low-level heuristics contains all the n possible RLS$_k$ operators, where n is the problem size. A by product of our analysis is the design of a more powerful variant of random gradient SHH that efficiently escapes from a larger variety of local optima.

Keywords: Hyper-heuristics, parameter adaptation, multimodal optimisation, runtime analysis, theory

1. Introduction

Hyper-heuristics (HHs) are automated algorithm design methodologies that aim to evolve efficient algorithms and related parameter values for the optimisation problem at hand [1, 2, 3]. HHs are generally subdivided into Generative HHs which aim to evolve offline the new heuristic from components of pre-existing heuristics, and Selection HHs (SHHs) which work online by selecting directly during the optimisation process the most promising low-level heuristic from a set of available ones. While the theoretical analysis of HHs is still in its infancy, in recent years considerable progress has been made on the foundational understanding of SHHs [4].

*An extended abstract of this paper was published at GECCO 2025 [23].

Email addresses: 12212608@mail.sustech.edu.cn (Yuxuan Ma), olivetop@sustech.edu.cn (Pietro S. Oliveto), john.warwicker@kit.edu (John Alasdair Warwicker)

SHHs are composed of two separate components: (1) a *heuristic selection methodology* which consists of a learning mechanism to decide which heuristic is to be applied next; (2) a *move acceptance operator* to decide whether the newly created search point should be accepted. Progress in the rigorous performance analysis of SHHs has been made on both of these components.

Concerning move-acceptance operators, it has been shown that they can be very powerful at helping the SHH at escaping from local optima in multimodal optimisation. Indeed, it has been proved that a simple move-acceptance mechanism that, with some fixed probability $0 < p < 1$, switches between elitist selection and accepting any solution regardless of its fitness allows the so-called Move Acceptance Hyper-Heuristic (MAHH) to optimise the standard multimodal CLIFF_d benchmark function in expected time $\mathcal{O}(\frac{n^3}{d^2} + n \log n)$ [5, 6]. This implies that when the basin of attraction d of the local optima is very large (i.e., linear in the problem size), the MAHH matches the $\mathcal{O}(n \log n)$ bound also achieved by artificial immune systems (AISs) equipped with an ageing operator and a (1,λ) EA with self-adjusting population size [7, 8]. However, when the basin of attraction is small (i.e., constant), the MAHH's $\mathcal{O}(n^3)$ runtime is a logarithmic factor smaller than the best known bound for the AISs and is still much faster than any other known elitist or non-elitist algorithm. In addition, it has recently been shown that using the same move-acceptance mechanism with the standard bit mutation (SBM) operator also allows to speed-up the time to escape from the local optimum of the harder JUMP_m function up to a factor of $\left(\mathcal{O}(\frac{\log n}{m})\right)^m$ by allowing the algorithm to traverse the valley of lower objective values of decreasing fitness in several steps [9, 10]. While this result highlights the effectiveness of the move-acceptance selection mechanism at escaping from local optima by accepting solutions of lower fitness, for the deceptive basin of attraction of the JUMP function greater speed-ups can be achieved by directly jumping to the optimum. Indeed, the runtime is worse than the best known as fast mutation operators [11, 12] and algorithms using stagnation detection [13, 14] give speed-ups of the order $m^{-\Theta(m)}$, while the addition of crossover allows for speed-ups of at least $\mathcal{O}(n^{-1})$ [15]. Further speed-ups leading to expected runtimes of approximately order $e^{\Theta(k)}(n/k)^{k/2}$ can be achieved by a theory-driven $(1 + (\lambda, \lambda))$ GA [16] and runtimes as low as approximately $\mathcal{O}(n \log n + 4^k)$ may be achieved using problem tailored artificial diversity enforcing mechanisms that increase the probability that a diverse population is present on the plateau, thus that the crossover operator can quickly create a globally optimal solution by recombining two locally optimal ones [17].

Concerning heuristic selection methodologies, considerable improvements have been made towards the understanding of the Generalised Random Gradient hyper-heuristic (GRG). This simple SHH chooses a low-level heuristic uniformly at random and continues to apply it so long as it finds improvements within a period of τ steps called the *learning period*. Otherwise, a new low-level heuristic is chosen, again uniformly at random. It has been shown that for the LEADINGONES benchmark function, GRG equipped with the set of Randomized Local Search (RLS) low-level heurstics $H = \{\text{RLS}_1, \ldots, \text{RLS}_k\}$, any $k = \Theta(1)$, matches the best possible expected runtime achievable (up to lower order terms) by any unbiased (1+1) black box algorithm using the same set of heuristics [18]. In particular, if equipped with at least the first segment of 18 operators, i.e., $H = \{\text{RLS}_1, \ldots, \text{RLS}_{18}\}$, GRG runs in an expected runtime of $\approx 0.388n^2$, the best possible runtime up to lower order terms of a (1+1) black box algorithm as calculated by Doerr and Wagner [19].

For GRG to achieve this performance, it is necessary that the learning period τ is set appropriately i.e., $\tau = \omega(n)$ and $\tau \leq (1/k - \epsilon)n \ln n$, $\epsilon > 0$ a constant [18]. Since the appropriate learning period for the optimisation problem at hand may be difficult to identify, an *adaptive ran-*

dom gradient hyper-heuristic was introduced that aims to avoid the burden of selecting the duration of the learning period in advance, but also adapts the value of τ during the run according to how efficient the individual low-level heuristics are at identifying improvements throughout the process [20]. Subsequent work has indeed shown that different functions require considerably different durations of the learning period for GRG to run in *optimal* expected runtime i.e., the best time achievable with the low-level heuristics [21]. For instance, while for the RIDGE (and LEADINGONES) benchmark τ should be super-linear in the problem size, for the ONEMAX function it should be sub-linear even if GRG is equipped with the best two low-level randomised local search heuristics for the problem i.e., RLS_1 and RLS_3. On the other hand, ARG can adapt the learning period effectively to guarantee optimal asymptotic performance for the three unimodal functions without having to choose τ in advance.

All previous work for GRG considered unimodal functions. In this paper, we extend the analysis of GRG to multimodal optimisation to evaluate the SHH's performance at escaping from local optima. For this purpose, we will consider the TWORATES function that was especially designed to evaluate the performance of a recently introduced self-adaptive algorithm called *flex-EA* [22]. Although not explicitly mentioned, the algorithm is also essentially a HH that uses both stagnation detection and an archive to keep track of the RLS_k operators that were successful in the past to select the next operator to be applied. The TWORATES function consists of a ONEMAX slope interrupted by an area of size $\sqrt{n}$ with a range of consecutive local optima separated by gaps of two alternating different widths, $\log n$ and $2\log n$. It was designed to evaluate the effectiveness of the archive strategy at storing and re-applying the most suitable RLS_k operators to make the two jumps of different lengths. The authors show that by saving the two identified operators for each gap size in the archive, the *flex-EA* optimises TWORATES in an expected runtime $\mathcal{O}(n^{4.5})$ and that it is faster by a poly-logarithmic factor than using the power-law mutation operator and by a super-polynomial factor compared to stagnation detection, i.e., the best known mutation-based algorithms for JUMP_m [13, 14, 11, 12].

In this paper, we evaluate the performance of the much simpler GRG SHH at tackling the two differently-sized gaps. Since GRG essentially only has an archive (or memory) of size 1, it cannot remember the two operators that have been more useful in the recent past, but can only save the information about the last operator that proved to be useful. Nevertheless, we show that GRG is even faster at crossing the range of local optima by being able to identify the operator that minimises the expected time to escape from the hardest of the two types of local optima and by continuing to apply it consecutively until they are all overcome. To evaluate the power of GRG at overcoming the "gapped area", we consider a simplified function which we call CUTTWORATES. We show that if the required operators are known, thus the heuristic set is small, i.e., $\Theta(1)$, then GRG can overcome the hurdles in expected time $\mathcal{O}(\frac{n^{3.67}}{\sqrt{\log n}})$. However, we also show that GRG does not obtain an asymptotic speed up by consecutively applying the optimal heuristic compared to selecting a new heuristic at each step uniformly at random in this setting. On the other hand, in realistic scenarios where the set of required heuristics is not known, then the heuristic set will contain a large number of low-level heuristics. Our first result is to show that GRG can achieve a speed-up of order $\Omega(\frac{\sqrt{n}}{\log^{5/2} n})$ compared to random heuristic selection on the expected time to cross the gapped area when equipped with a linear number of heuristics.

Concerning the complete TWORATES function, GRG equipped with a super-constant number of heuristics may struggle more on the final ONEMAX part when hillclimbing towards the opti-

Algorithm 1 Generalised Random Gradient Hyper-heuristic [18]

```
1: Choose $x \in S$ uniformly at random
2: while stopping conditions not satisfied do
3:     Choose $h \in H$ uniformly at random
4:     $c_t \leftarrow 0$
5:     while $c_t < \tau$ do
6:         $c_t \leftarrow c_t + 1$; $x' \leftarrow h(x)$
7:         if $f(x') > f(x)$ then
8:             $c_t \leftarrow 0$; $x \leftarrow x'$
```

mum, rather than in the "gapped region". Since the learning period needs to be set to a pretty large value, long enough for the optimal operator to perform the jumps, i.e., approx. $\tau \geq n^{3.17}$, sub-optimal operators may be applied during the hillclimbing phases and be successful within τ steps for a long time, impacting the overall expected runtime. Nevertheless, we show that GRG is still faster than all previously studied algorithms for the original TWORATES function by providing an upper bound of $\mathcal{O}(\tau n \log n)$ for polynomial $\tau \geq n^{\log 9+\epsilon}$ on its expected time when equipped with all operators i.e., $H = \{\text{RLS}_1, \ldots, \text{RLS}_n\}$. This is $\mathcal{O}(n^{4.17} \log n)$ for $\tau = n^{3.17}$. Compared to previous analyses of SHHs, this is the first time super-constant low-level heuristic set sizes are considered up to the complete set consisting of all n different neighbourhood sizes for RLS$_k$.

Since our results rely heavily upon the fact that the TWORATES function is designed to have both gaps of even length, we conclude the paper with a discussion of how to generalise GRG to continue using the same operator to overcome multiple optima of both odd and even lengths; thus, how to design a more powerful GRG SHH for multimodal optimisation.

The rest of the paper is structured as follows. In Section 2, we introduce the GRG SHH and the multimodal optimisation problems we consider (i.e., TWORATES and CUTTWORATES). In Section 3, we identify the best heuristic to allow GRG to efficiently escape the local optima before analysing GRG equipped respectively with best two operators in Section 4 and with the complete heuristic set in Section 5. We finish with some conclusions and avenues for future work.

Compared to its extended abstract [23], this paper has been considerably improved. Several proofs were originally missing. Others now include many previously omitted steps, while the proofs of lemmata 5 and 6 have been considerably improved and made more elegant.

2. Preliminaries

Throughout, we use the term RLS$_m$ to denote the randomised local search heuristic which samples a new solution with Hamming distance m to the current solution. We use $\log$ to denote the logarithm function with base 2.

2.1. The Random Gradient Hyper-heuristic

The Generalised Random Gradient hyper-heuristic (GRG) is shown in Algorithm 1. Let S be a finite search space, H a set of low-level heuristics and $f : S \to \mathbb{R}$ a fitness function. GRG selects a heuristic uniformly at random from the heuristic set H and applies it for τ steps unless an improvement is identified before. If so, the learning period counter τ is reset to 0, and the operator continues to be applied. Otherwise, a new operator is selected from H at random and a new learning phase of τ steps is started.

The algorithm was first analysed as the Random Gradient HH (see e.g., [24]) with a learning period of $\tau = 1$ by Alanazi and Lehre for the well-studied LEADINGONES benchmark function when equipped with $H = \{\text{RLS}_1, \text{RLS}_2\}$, randomised local search algorithms respectively with neighbourhood sizes of 1-bit and 2-bits [25][1]. However, for such a short learning period, the SHH performs equivalently to a Simple Random SHH that chooses a different operator at random at each step. The term *Generalised* Random Gradient was introduced by Lissovoi et al. when they analysed the effects of allowing arbitrarily long learning periods of duration τ, which lead the GRG to optimise LEADINGONES in *optimal* expected runtime up to the leading constant when equipped with $H = \{\text{RLS}_1, \dots, \text{RLS}_k\}$ for any constant k if τ is chosen appropriately [18], or if it is self-adapted in the Adaptive Random Gradient HH [20]. The analysis was also extended to other unimodal benchmark functions [21].

2.2. Multimodal Functions

The multimodal optimisation function we consider is TWORATES [22]. Let $\text{ONEMAX}(x) := |x|_1$ denote the number of 1-bits in the current solution x (where $x \in \{0,1\}^n$), and let $s := (3/4)n - \sqrt{n}$, $(3/4)n$, $g := \log n$ and $\sqrt{n}/g$ all be integers. Then,

$$\text{TWORATES}(x) := \begin{cases} \text{OM}(x) & \text{if } |x|_1 < s \text{ or } |x|_1 \geq 3n/4 \\ & \text{ or } |x|_1 = s + ig \text{ for } i \in \{0, 2, 3, 5, 6, \dots\}; \\ -1 & \text{otherwise.} \end{cases}$$

The function consists of a ONEMAX slope i.e., $\text{OM}(x)$, leading to a HURDLE section of length $\sqrt{n}$ consisting of alternating gaps of length $g := \log n$ and $2g := 2\log n$ (see Figure 1). The function was designed to show that the archive used by the *Flex-EA* is capable of storing and maintaning the two RLS_k operators (i.e., RLS_g and RLS_{2g}) necessary to perform the two different types of jump. The authors prove an upper bound of $\mathcal{O}(n^{4.5})$ for their *flex-EA* which they also prove to be a polylogarithmic speed up of $\Omega((\log n)^{\beta - 1/2})$ over fast mutation [11], while stagnation detection [13, 14] requires a super polynomial time of $n^{\Omega(\log n)}$.

Our aim is to analyse whether GRG can take advantage of the fact that it repeatedly applies a successful operator to cross the HURDLE region even faster. The task presents two challenges.

Firstly, GRG would have to repeatedly apply the same operator to overcome both gap types i.e., of sizes $g = \log n$ and $2g = 2\log n$. Notice that the function was defined in [22] to have both gaps, g and $2g$ of even length. This holds since both $g = \log n$ and $\sqrt{n}/g$ are integers, and thus $\sqrt{n} = 2^{\frac{g}{2}}$ is an integer, so g must be even. This is crucial for GRG to be able to cross both kinds of hurdles without changing operator. In the conclusion, we will discuss how to generalise the GRG hyperheuristic to be able to repeatedly overcome local optima of both odd and even lengths without having to switch operator, when the gaps are not too different in size.

Secondly, the learning period should be long enough to allow the optimal operator to successfully perform each of the jumps before a new operator is selected. The required long learning periods imply that GRG may also struggle at hillclimbing the ONEMAX parts of the function.

[1]The analysis was performed by choosing the bits with replacement but the results also hold if selected without replacement.

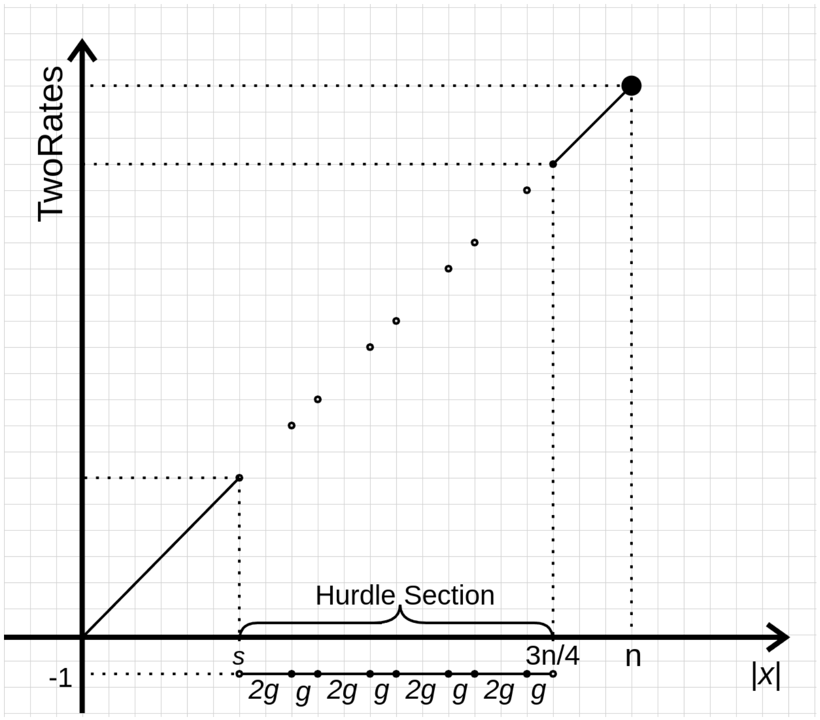

Figure 1: **The TWORATES fitness function**, where s denotes $3n/4 - \sqrt{n}$ and g denotes $\log_2 n$. Note that the graph magnifies the size of the hurdle region

Lissovoi et al. proved an $\omega(n \log n)$ runtime for GRG already when only equipped with the minimal heuristic set $H = \{\mathrm{RLS}_1, \mathrm{RLS}_3\}$ and $\tau \geq n$ [21]. The parameter setting necessary for TWORATES is more prohibitive. For GRG to successfully overcome the hurdles without changing the successful operator, the learning period τ has to be much larger than n, and the heuristic set should contain operators of at least logarithmic neighbourhood size. These, though, are inefficient in the hillclimbing phases, yet they may still improve if applied for a very large learning period.

With the aim of isolating the runtime of GRG to overcome the HURDLE part, we simplify the function as follows:

$$\text{CUTTWORATES}(x) := \begin{cases} \text{TWORATES}(x) & \text{if } |x|_1 \leq 3n/4; \\ -1 & \text{otherwise.} \end{cases}$$

The function is truncated to get insights into the behaviour of GRG up to the HURDLE region while considerably simplifying the analysis. Since the time to overcome the hurdles dominates the overall runtime for the previously considered algorithms, the simplified function allows us to initially ignore the final ONEMAX slope with RLS_k and super-constant k and provides the biggest possible speed-ups for GRG to overcome the hurdles compared to the runtime of the best known algorithms [22].

2.3. Notation

To estimate the probabilities of different operators to jump a hurdle, we use the same notation from [22]. We let $p(n, a, d, i)$ be the probability to reach a point of $a + d$ one-bits from a point with

a one-bits when flipping $d + 2i$ bits. It is necessary that $d + i$ zero-bits and i one-bits flip to reach such a point. Thus,

$$p(n, a, d, i) = \binom{n-a}{d+i}\binom{a}{i}/\binom{n}{d+2i}.$$

The following bound for hypergeometrically distributed random variables can be derived directly from [26], by substituting M by $N - M$ and i by $n - i$.

Lemma 1. *Let $X \sim \mathcal{H}(N, M, n)$ be a hypergeometrically distributed random variable (i.e. $\Pr[X = i] = \binom{M}{i}\binom{N-M}{n-i}\binom{N}{n}^{-1}$). Then for any t such that $0 < t < M/N$, the following inequality holds:*

$$\Pr[X \le (M/N - t)n] \le \exp(-2t^2 n).$$

3. The Best Operator for the Hurdles

In this section, we show that the best operator to be applied consecutively by GRG to cross the hurdle region is $\mathrm{RLS}_{4g} := \mathrm{RLS}_{4\log n}$. We first provide a helper lemma that provides bounds on the binomial coefficient using Stirling's approximation.

Lemma 2. *For $s > t > 0$,*

$$\frac{(s-t)^t}{t^{t+1/2}}\frac{e^{t-\frac{1}{12t}}}{\sqrt{2\pi}} < \binom{s}{t} < \frac{s^t}{t^{t+1/2}}\frac{e^{t+\frac{1}{360t^3}-\frac{1}{12t}}}{\sqrt{2\pi}}.$$

Proof. By Stirling's Approximation, we have:

$$\sqrt{2\pi}n^{n+1/2}\exp\left[\frac{1}{12n} - \frac{1}{360n^3} - n\right] < \Gamma(n+1) < \sqrt{2\pi}n^{n+1/2}\exp\left[\frac{1}{12n} - n\right].$$

Thus we get,

$$\binom{s}{t} > \frac{(s-t)^t}{\Gamma(t+1)} > (s-t)^t t^{-t-1/2}\exp\left[t - \frac{1}{12t}\right]/\sqrt{2\pi}.$$

In the other direction we get,

$$\binom{s}{t} < \frac{s^t}{\Gamma(t+1)} < s^t t^{-t-1/2}\exp\left[t + \frac{1}{360t^3} - \frac{1}{12t}\right]/\sqrt{2\pi}.$$

□

Now we prove the main result of this section, which is to identify the fastest heuristic of the form $\mathrm{RLS}_{\rho g} := \mathrm{RLS}_{\rho\log n}$ to cross the hurdles region of the TwoRates function i.e., the best single value for ρ.

Theorem 1. *Let $\rho > \gamma > 0$ be some constants, and let $\beta = \rho/\gamma$. Then the probability of using $RLS_{\rho g}$ and getting an improvement of γg in the hurdles region of the* TwoRates *function is equal to*

$$p(n, a, \gamma g, (\rho-\gamma)g/2) = (1 - o(1))\sqrt{2\rho/(\pi(\rho^2-\gamma^2)\log n)}\,n^{-\gamma f(\beta)},$$

where $f(\beta) = \beta[1 + \log\sqrt{(\beta^2-1)/(3\beta^2)}] + \log\sqrt{3(\beta+1)/(\beta-1)}$.

Proof. We aim to derive tight upper and lower bounds for the probability

$$p(n, a, \gamma g, (\rho - \gamma)g/2) = \binom{n-a}{(\rho+\gamma)g/2}\binom{a}{(\rho-\gamma)g/2}/\binom{n}{\rho g}$$

We first develop the lower bound. Note that $\rho g = \Theta(\log n)$, $3n/4 - \sqrt{n} \leq a \leq 3n/4$ and thus $n/4 \leq n - a \leq n/4 + \sqrt{n}$. We bound the probability term by term using Lemma 2:

$$\binom{n-a}{(\rho+\gamma)g/2} > \frac{(n/2)^{(\rho+\gamma)g/2}(1-2(\rho+\gamma)g/n)^{(\rho+\gamma)g/2}\exp\left[\frac{(\rho+\gamma)g}{2} - \frac{1}{6(\rho+\gamma)g}\right]}{((\rho+\gamma)g)^{(\rho+\gamma)g/2}\sqrt{\pi(\rho+\gamma)g}};$$

$$\binom{a}{(\rho-\gamma)g/2} > \frac{(3n/2)^{(\rho-\gamma)g/2}(1-(4\sqrt{n}+2(\rho-\gamma)g)/(3n))^{(\rho-\gamma)g/2}\exp\left[\frac{(\rho-\gamma)g}{2} - \frac{1}{6(\rho-\gamma)g}\right]}{((\rho-\gamma)g)^{(\rho-\gamma)g/2}\sqrt{\pi(\rho-\gamma)g}};$$

$$1/\binom{n}{\rho g} > \frac{(\rho g)^{\rho g}\sqrt{2\pi\rho g}\exp\left[-\rho g - \frac{1}{360(\rho g)^3} + \frac{1}{12\rho g}\right]}{n^{\rho g}}.$$

To prove the lower bound of

$$p(n, a, \gamma g, (\rho-\gamma)g/2) > \sqrt{\frac{2\rho}{\pi(\rho^2-\gamma^2)}}\frac{\kappa_1(n)}{n^{\gamma f(\beta)}\sqrt{\log n}},$$

where

$$\kappa_1(n) = \left(1 - \frac{2(\rho+\gamma)g}{n}\right)^{(\rho+\gamma)g/2}\left(1 - \frac{4\sqrt{n}+2(\rho-\gamma)g}{3n}\right)^{(\rho-\gamma)g/2}$$
$$\cdot \exp\left[-\frac{1}{6(\rho+\gamma)g} - \frac{1}{6(\rho-\gamma)g} - \frac{1}{360(\rho g)^3} + \frac{1}{12\rho g}\right].$$

we multiply the three bounded terms. Firstly, we see that,

$$\frac{\sqrt{2\pi\rho g}}{\sqrt{\pi(\rho+\gamma)g}\sqrt{\pi(\rho-\gamma)g}} = \sqrt{\frac{2\rho}{\pi(\rho^2-\gamma^2)\log n}},$$

and, using the fact that $(a+b)/2 > (a^{-1}+b^{-1})^{-1}$ when $a, b > 0$, we further have that

$$\frac{1}{12\rho g} < \frac{2}{6(\rho+\gamma)g + 6(\rho-\gamma)g} < \frac{1}{6(\rho+\gamma)g} + \frac{1}{6(\rho-\gamma)g}.$$

Thus, $\kappa_1(n) = 1 - o(1)$. Secondly, we will show that,

$$\frac{(n/2)^{(\rho+\gamma)g/2}}{((\rho+\gamma)g)^{(\rho+\gamma)g/2}}\frac{(3n/2)^{(\rho-\gamma)g/2}}{((\rho-\gamma)g)^{(\rho-\gamma)g/2}}\frac{(\rho g)^{\rho g}}{n^{\rho g}} = \frac{1}{n^{\gamma f(\beta)}},$$

where $f(\beta)$ is defined as in the theorem statement. We re-group the terms as follows into two multiplicative terms, which we denote A and B (note that $g := \log_2 n$):

$$A \cdot B := \frac{(n/2)^{(\rho+\gamma)g/2}(3n/2)^{(\rho-\gamma)g/2}}{n^{\rho g}} \cdot \frac{(\rho g)^{\rho g}}{((\rho+\gamma)g)^{(\rho+\gamma)g/2}((\rho-\gamma)g)^{(\rho-\gamma)g/2}},$$

We can simplify the first multiplicative term A as follows:

$$\begin{aligned}
A &= \frac{(n/2)^{(\rho+\gamma)g/2}(3n/2)^{(\rho-\gamma)g/2}}{n^{\rho g}} \\
&= \frac{n^{(\rho+\gamma)g/2}n^{(\rho-\gamma)g/2}}{n^{\rho g}}\left(\frac{1}{2}\right)^{(\rho+\gamma)g/2}\left(\frac{3}{2}\right)^{(\rho-\gamma)g/2} \\
&= 1 \cdot \frac{1}{n^{(\rho+\gamma)/2}}\frac{n^{(\log 3)(\rho-\gamma)/2}}{n^{(\rho-\gamma)/2}} \\
&= \frac{n^{\log\sqrt{3}(\rho-\gamma)}}{n^{\rho}} = \frac{n^{\log\sqrt{3}\gamma(\beta-1)}}{n^{\gamma\beta}},
\end{aligned}$$

and simplify the second multiplicative term B as follows:

$$\begin{aligned}
B &= \frac{(\rho g)^{\rho g}}{((\rho+\gamma)g)^{(\rho+\gamma)g/2}((\rho-\gamma)g)^{(\rho-\gamma)g/2}} \\
&= \frac{\rho^{\rho g}}{(\rho+\gamma)^{(\rho+\gamma)g/2}(\rho-\gamma)^{(\rho-\gamma)g/2}}\frac{g^{\rho g}}{g^{(\rho+\gamma)g/2}g^{(\rho-\gamma)g/2}} \\
&= \frac{\rho^{(\rho+\gamma)g/2}\rho^{(\rho-\gamma)g/2}}{(\rho+\gamma)^{(\rho+\gamma)g/2}(\rho-\gamma)^{(\rho-\gamma)g/2}} \cdot 1 \\
&= \frac{1}{(1+1/\beta)^{(\rho+\gamma)g/2}}\frac{1}{(1-1/\beta)^{(\rho-\gamma)g/2}} \\
&= \frac{1}{n^{\frac{\rho+\gamma}{2}\log(1+1/\beta)}}\frac{1}{n^{\frac{\rho-\gamma}{2}\log(1-1/\beta)}} \\
&= \frac{1}{n^{\gamma\frac{\beta+1}{2}\log\frac{\beta+1}{\beta}}n^{\gamma\frac{\beta-1}{2}\log\frac{\beta-1}{\beta}}}.
\end{aligned}$$

Thus, we have

$$A \cdot B := n^{-\gamma\left(\beta-(\beta-1)\log\sqrt{3}+\frac{\beta+1}{2}\log\frac{\beta+1}{\beta}+\frac{\beta-1}{2}\log\frac{\beta-1}{\beta}\right)}.$$

We simplify the term in the exponent as follows:

$$\begin{aligned}
&\beta-(\beta-1)\log\sqrt{3}+\frac{\beta+1}{2}\log\frac{\beta+1}{\beta}+\frac{\beta-1}{2}\log\frac{\beta-1}{\beta} \\
&= \beta-\beta\log\sqrt{3}+\log\sqrt{3}+\beta\log\sqrt{\frac{\beta+1}{\beta}}+\log\sqrt{\frac{\beta+1}{\beta}}+\beta\log\sqrt{\frac{\beta-1}{\beta}}-\log\sqrt{\frac{\beta-1}{\beta}} \\
&= \beta\left(1-\log\sqrt{3}+\log\sqrt{\frac{\beta+1}{\beta}}+\log\sqrt{\frac{\beta-1}{\beta}}\right) \\
&\quad + \left(\log\sqrt{3}+\log\sqrt{\frac{\beta+1}{\beta}}-\log\sqrt{\frac{\beta-1}{\beta}}\right) \\
&= \beta\left(1+\log\sqrt{\frac{\beta^2-1}{3\beta^2}}\right)+\log\sqrt{\frac{3(\beta+1)}{\beta-1}} \\
&:= f(\beta).
\end{aligned}$$

Hence,

$$p(n,a,\gamma g,(\rho-\gamma)g/2) > \sqrt{\frac{2\rho}{\pi(\rho^2-\gamma^2)}}\frac{\kappa_1(n)}{n^{\gamma f(\beta)}\sqrt{\log n}} = (1-o(1))\sqrt{\frac{2\rho}{\pi(\rho^2-\gamma^2)\log n}}n^{-\gamma f(\beta)}.$$

The upper bound follows similar calculations:

$$\binom{n-a}{(\rho+\gamma)g/2} < \frac{(n/2)^{(\rho+\gamma)g/2}(1+4/\sqrt{n})^{(\rho+\gamma)g/2}\exp\left[\frac{(\rho+\gamma)g}{2}+\frac{1}{45(\rho+\gamma)^3g^3}-\frac{1}{6(\rho+\gamma)g}\right]}{((\rho+\gamma)g)^{(\rho+\gamma)g/2}\sqrt{\pi(\rho+\gamma)g}};$$

$$\binom{a}{(\rho-\gamma)g/2} < \frac{(3n/2)^{(\rho-\gamma)g/2}\exp\left[\frac{(\rho-\gamma)g}{2}+\frac{1}{45(\rho-\gamma)^3g^3}-\frac{1}{6(\rho-\gamma)g}\right]}{((\rho-\gamma)g)^{(\rho-\gamma)g/2}\sqrt{\pi(\rho-\gamma)g}};$$

$$1/\binom{n}{\rho g} < \frac{(\rho g)^{\rho g}\sqrt{2\pi\rho g}\exp\left[-\rho g+\frac{1}{12\rho g}\right]}{n^{\rho g}(1-\rho g/n)^{\rho g}}.$$

Combining these terms together, we get,

$$p(n,a,\gamma g,(\rho-\gamma)g/2) < \sqrt{\frac{2\rho}{\pi(\rho^2-\gamma^2)}}\frac{\kappa_2(n)}{n^{\gamma f(\beta)}\sqrt{\log n}},$$

where

$$\begin{aligned}\kappa_2(n) &= (1+4/\sqrt{n})^{(\rho+\gamma)g/2}(1-\rho g/n)^{-\rho g}\\ &\cdot\exp\left[\frac{1}{45(\rho+\gamma)^3g^3}+\frac{1}{45(\rho-\gamma)^3g^3}+\frac{1}{12\rho g}-\frac{1}{6(\rho+\gamma)g}-\frac{1}{6(\rho-\gamma)g}\right],\end{aligned}$$

which is $1-o(1)$. This concludes the proof. □

Corollary 1. *The value of m, $1 \le m \le n$, that minimises the expected runtime of* RLS_m *for crossing the* HURDLE *region of* TWORATES *starting from $|x|_1 = 3n/4-\sqrt{n}$, converges to $m = 4g$ (i.e., $m = 4\log n$) when n is sufficiently large.*

Proof. Once we fix an operator RLS_m and the current ONEMAX value which is a, we have

$$p(n,a,m-2i,i) = \binom{n-a}{m-i}\binom{a}{i}/\binom{n}{m},$$

where $i \sim \mathcal{H}(n,a,m)$. Let $\Pr_m[d=\hat{d}]$ denote the probability that our chosen RLS_m heuristic finds an improvement of size $\hat{d}$ in the hurdles region of TWORATES. According to Lemma 1, we have:

$$\Pr_m[d\ge 2g] \le \Pr_m[d\ge 0] = \Pr_m[i\le m/2] \le \exp[-2(a/n-1/2)^2m],$$

where $3n/4-\sqrt{n} < a < 3n/4$. Thus, if $m=\omega(\log n)$, we get that the probability for RLS_m to make any improvement is $1/n^{\omega(1)}$, which is smaller than the probability for the operators RLS_k to make an improvement of $2g$ where $k=\Theta(\log n)$.

From Theorem 1 we know that,

$$p(n, a, \gamma g, (\rho - \gamma)g/2) = (1 - o(1))\sqrt{2\rho/(\pi(\rho^2 - \gamma^2)\log n)}n^{-\gamma f(\beta)}.$$

When n is sufficiently large, the term $n^{-\gamma f(\beta)}$ dominates the whole expression, and one can check that

$$f'(\beta) = 1 + \log\sqrt{\frac{x^2 - 1}{3x^2}}$$

is monotonically increasing when $\beta > 1$ and $f'(\beta) = 0$ if and only if $\beta = 2$ when $\beta > 1$. Thus, for $\beta > 1$, $f(\beta)$ reaches its minimum at $\beta = \rho/\gamma = 2$. For the hardest hurdle with length $2g$ (i.e., we fix $\gamma = 2$), we get that $\rho = 4$ maximises the term $n^{-\gamma f(\beta)}$. Since the time to cross the $2g$ hurdle dominates that of crossing the easier hurdle, we conclude that the optimal number of bits to flip during the HURDLE region converges to $4g$ when n is sufficiently large. □

4. Using the Two Optimal Operators

In this section, we analyse the performance of GRG on the functions CUTTWORATES and TWORATES when equipped with the two best operators for the ONEMAX and HURDLE regions (i.e., RLS_1 and RLS_{4g} respectively). We compare these results with the performance of the Simple Random HH, which at each iteration chooses a new operator i.e., essentially GRG with $\tau = 0$.

We first note the following result which follows directly from the definition of $p(n, a, d, i)$ (see e.g., [27]).

Lemma 3. *The probability that* RLS_k *(*$1 \leq k \leq n$*) makes any improvement for* ONEMAX *decreases with the number of one-bits in the bit string x for $n/2 \leq |x|_1 \leq n$.*

We also note the following loose upper bound on the expected time for GRG to optimise ONEMAX when it contains the optimal operator for this function (i.e., RLS_1).

Lemma 4. *The expected runtime of GRG on* ONEMAX *with any operator set H where* $\text{RLS}_1 \in H$ *is* $\begin{cases} \mathcal{O}(|H|\tau n \log n) & \text{if } \tau = o(n), \\ \mathcal{O}(|H|\tau n) & \text{if } \tau = \Omega(n). \end{cases}$

Proof. We first look at the time for the algorithm to make one improvement. We will assume the improvement is made by RLS_1 and that when any other operator is chosen, which happens with probability $(1-1/|H|)$, we pessimistically assume that they fail within τ steps (or the improvement is found before). RLS_1 is chosen with probability $1/|H|$. Let P_F denote the probability that RLS_1 fails to find an improvement within τ steps. If this event occurs, we wait for RLS_1 to be chosen again. Otherwise, with probability $1 - P_F$, RLS_1 improves within τ steps. The absorbing time of the described Markov process is an upper bound on the time T_s to improve by one bit. So,

$$\begin{aligned} \mathbb{E}[T_s] &= \frac{1}{|H|}\Big(P_F(\tau + \mathbb{E}[T_s]) + (1 - P_F)2\tau\Big) + \frac{|H| - 1}{|H|}(\tau + \mathbb{E}[T_s]) \\ &= 2\tau + \frac{P_F + |H| - 1}{1 - P_F}\tau, \end{aligned}$$

where the 2τ comes from pessimistically assuming that once RLS_1 improves, it fails to improve again within the next τ steps.

Thus, for $\tau = \Omega(n)$ we get,

$$\mathbb{E}[T_s] = 2\tau + \frac{P_F + |H| - 1}{1 - P_F}\tau < 2\tau + \frac{|H|}{1 - P_F}\tau < 2\tau + \frac{\mathrm{e}|H|}{\mathrm{e} - 1}\tau = \mathcal{O}(|H|\tau),$$

where we used that the probability of improving by RLS_1 is always at least $1/n$ to bound $P_F \le (1 - 1/n)^\tau \le 1/e$.

Since at most n bits have to improve, multiplying the upper bound on the time to improve one bit by n provides a bound on the expected time to hit the optimum of $O(|H|\tau n)$ which proves the second statement.

For the case $\tau = o(n)$, we use that when the current solution has $n - i$ 1-bits, the probability of improving by RLS_1 is i/n to bound $P_F = (1 - \frac{i}{n})^\tau \le 1 - \frac{i}{n}$. So the expected time to improve upon fitness level $n - i$ is,

$$\mathbb{E}[T_s \mid \text{OM} = n - i] < 2\tau + \frac{|H|}{1 - P_F}\tau < 2\tau + \frac{n}{i}|H|\tau,$$

and by summing up over all fitness levels we get the first claim,

$$\mathbb{E}[T_s] = \sum_{i=1}^{n} \mathbb{E}[T_s \mid \text{OM} = n - i] < 2n\tau + |H|\tau \sum_{i=1}^{n} \frac{n}{i} = \mathcal{O}(|H|\tau n \log n). \qquad \square$$

Now we derive a general bound on the expected runtime of the Simple Random HH for TWORATES.

Theorem 2. *The expected time for the Simple Random Hyper-heuristic, equipped with any operator set H where $\text{RLS}_1, \text{RLS}_{4g} \in H$, to optimise* TWORATES *is $\mathcal{O}(|H| n^{\log 9}\sqrt{n/\log n})$.*

Proof. First, we derive an upper bound for the expected runtime of the Simple Random HH on the HURDLE region. Assume pessimistically that any operator except RLS_{4g} does not make any improvement. Then, denote by T_r the absorbing time for the described Markov process for any $2g$ hurdle. Let P_f denote the probability that RLS_{4g} fails to find an improvement in a single step. Then

$$\mathbb{E}[T_r] = \frac{1}{|H|}\Big(P_f(1 + \mathbb{E}[T_r]) + 1 - P_f\Big) + \frac{|H| - 1}{|H|}(1 + \mathbb{E}[T_r]).$$

So,

$$\mathbb{E}[T_r] = 1 + \frac{|H| - 1 + P_f}{1 - P_f} < 1 + \frac{|H|}{1 - P_f}.$$

According to Theorem 1, we have $1 - P_f = (1 - o(1))\sqrt{2/(3\pi \log n)}n^{-\log 9}$ for the $2g$ hurdle, so $\mathbb{E}[T_r] = \mathcal{O}(|H| n^{\log 9}\sqrt{\log n})$. For the hurdle of size g, we have $1 - P_f = (1 - o(1))\sqrt{8/(15\pi \log n)}n^{-\log(25\sqrt{5}/16)}$, which is the larger term and thus the value $\mathbb{E}[T_r]$ for this case is smaller. Since there are $\frac{2\sqrt{n}}{3\log n}$ hurdles, the whole expected runtime for the Simple Random Hyper-heuristic to jump the HURDLE region is $\mathcal{O}(|H| n^{\log 9}\sqrt{n/\log n})$.

Now, we bound the expected runtime for the initial and final ONEMAX slopes. We will directly use the expected runtime for the simple random HH on the whole ONEMAX function as the upper bound. We denote by $\mathbb{E}[T_{om}|\text{OM} = n-i]$ the expected time for the simple random HH to improve on the ONEMAX function when at OM $= n-i$ 1-bits. Then, with similar calculations to those of Lemma 4, we get,

$$\mathbb{E}[T_{om}] = \frac{1}{|H|}\Big(\frac{n-i}{n}(1+\mathbb{E}[T_{om}]) + \frac{i}{n}\Big) + \frac{|H|-1}{|H|}(1+\mathbb{E}[T_{om}]),$$

and, $\mathbb{E}[T_{om} \mid \text{OM} = n-i] < 1 + \frac{|H|n}{i}$. Thus, summing up over all n fitness levels, we get,

$$\mathbb{E}[T_{om}] = n + |H|\sum_{i=1}^{n}\frac{n}{i} = \mathcal{O}(|H|n\log n).$$

Thus, we conclude that the expected time for the Simple Random HH to optimise TWORATES is $\mathcal{O}(|H|n^{\log 9}\sqrt{n/\log n})$. □

We now consider GRG equipped with the best two operators i.e., $H = \{\text{RLS}_1, \text{RLS}_{4g}\}$ to identify the best possible runtime achievable by the HH for CUTTWORATES. The operators are chosen because RLS_1 is the best possible operator for the ONEMAX part and we derived in the previous section that RLS_{4g} is the best operator for the hurdles part.

Theorem 3. *Let ϵ be an arbitrarily small positive constant. The expected runtime of the Generalised Random Gradient hyper-heuristic for* CUTTWORATES *with $H = \{RLS_1, RLS_{4g}\}$ and $\tau \geq n^{\log 9+\epsilon}$, $\tau = poly(n)$ is $\mathcal{O}(\tau + n^{\log 9}\sqrt{n/\log n})$.*

Proof. We separate the whole runtime into two stages: the ONEMAX stage (i.e., when the current CUTTWORATES value is less than or equal to $3n/4 - \sqrt{n}$), and the HURDLE stage (i.e., when the current CUTTWORATES value is between $3n/4 - \sqrt{n}$ and $3n/4$). Let T_1 and T_2 denote the time between the first random operator choice in each stage of the optimisation process and the moment when the necessary fitness value to enter the subsequent stage is reached (or the global optimum is found). Let S denote the number of iterations between the first solution in stage 2 being constructed and the first operator choice occurring in stage 2. Then, the expected runtime $\mathbb{E}[T]$ can be bounded by $\mathbb{E}[T] < \mathbb{E}[T_1] + \mathbb{E}[T_2] + \mathbb{E}[S]$.

We first give an upper bound for $\mathbb{E}[T_1]$. Let $\mathbb{E}[T_1 \mid \text{RLS}_i]$ denote the expected time to complete the first stage if RLS_i is the first randomly chosen operator. Since we choose operators u.a.r., we have:

$$\mathbb{E}[T_1] = \frac{1}{2}\mathbb{E}[T_1 \mid \text{RLS}_1] + \frac{1}{2}\mathbb{E}[T_1 \mid \text{RLS}_{4g}].$$

Let $\mathcal{A}_i$ denote the event that once RLS_i is selected, it never fails during the current stage. Then, using the law of total expectation, we have:

$$\mathbb{E}[T_1 \mid \text{RLS}_1] = \mathbb{E}[T_1 \mid \text{RLS}_1, \mathcal{A}_1]\Pr[\mathcal{A}_1] + \mathbb{E}[T_1 \mid \text{RLS}_1, \overline{\mathcal{A}_1}]\Pr[\overline{\mathcal{A}_1}].$$

Since $\tau \geq n^{\log 9+\epsilon}$ and the probability for RLS_1 to make an improvement during each iteration is greater than $\frac{3n/4-\sqrt{n}}{n}$, we can bound $\Pr[\overline{\mathcal{A}_1}]$ from above by

$$\Pr[\overline{\mathcal{A}_1}] < 1 - \left(1 - \left(1 - \frac{3n/4-\sqrt{n}}{n}\right)^{\tau}\right)^{n/4-\sqrt{n}} < 1 - \left(1 - \exp\left[-n^{\log 9}\right]\right)^{n}.$$

This inequality holds since the probability for RLS_1 to make an improvement within τ iterations is greater than $1-(1-(3n/4-\sqrt{n})/n)^\tau$ and thus greater than $1-(\frac{1}{4}+\frac{1}{\sqrt{n}})^{n^{\log 9}} > 1-\exp[-n^{\log 9}]$ when $n \geq 16$. Further, with high probability, it has to make at most $n/4+o(n)-\sqrt{n} < n$ consecutive improvements if we start from a random bit string (noting an expected runtime of $\mathcal{O}(n\tau)$ if this does not occur via Lemma 4, giving an additional $o(\tau)$ term in our final result). Thus, we have:

$$\Pr[\overline{\mathcal{A}}_1] < 1-\exp\left[-\frac{n}{\exp[n^{\log 9}]-1}\right] < \frac{n}{\exp[n^{\log 9}]-1} < \frac{n}{\exp[n^3]}.$$

The first inequality holds because we have $(1-1/x)^{x-1} > 1/\mathrm{e}$, while the second inequality holds because $1-\exp[-x] \leq x$. The last inequality holds because $\log 9 > 3$.

By Lemma 4, we will get $\mathbb{E}[T_1 \mid \mathrm{RLS}_1, \overline{\mathcal{A}}_1] = \mathcal{O}(n\tau)$. This holds because once $\overline{\mathcal{A}}_1$ happens, its expected runtime cannot be worse than that of the assumption we make when proving Lemma 4.

Since we know that $\mathbb{E}[T_1 \mid \mathrm{RLS}_1, \mathcal{A}_1] = \mathcal{O}(n\log n)$ [28], and $\Pr[\mathcal{A}_1] < 1$, we can conclude that

$$\mathbb{E}[T_1 \mid \mathrm{RLS}_1] < \mathcal{O}(n\log n)\cdot 1 + \mathcal{O}(n\tau)\cdot\frac{n}{\exp[n^3]} = \mathcal{O}(n\log n).$$

We now seek an upper bound for $\mathbb{E}[T_1 \mid \mathrm{RLS}_{4g}]$.

Let $\Pr_{4g}[d=\hat{d}]$ be the probability that RLS_{4g} makes an improvement of $\hat{d}$ at $|x|_1 = \frac{3n}{4}-\sqrt{n}$. Then $\Pr_{4g}[d>0] > \Pr_{4g}[d=2]$. Then we use the same bound strategy showed in the proof of Theorem 1. We have that:

$$\Pr_{4g}[d=2] = \tbinom{n/4+\sqrt{n}}{2g+1}\tbinom{3n/4-\sqrt{n}}{2g-1}/\tbinom{n}{4g},$$

where we bound

$$\binom{n/4+\sqrt{n}}{2g+1} > \frac{(n/4)^{2g+1}(1+\frac{4(\sqrt{n}-2g-1)}{n})^{2g+1}\exp[2g+1-\frac{1}{12(2g+1)}]}{(2g)^{2g+1}(1+1/(2g))^{2g+1}\sqrt{2\pi(2g+1)}};$$

$$\binom{3n/4-\sqrt{n}}{2g-1} > \frac{(3n/4)^{2g-1}(1-\frac{4(\sqrt{n}+2g-1)}{3n})^{2g-1}\exp[2g-1-\frac{1}{12(2g-1)}]}{(2g)^{2g-1}(1-1/(2g))^{2g-1}\sqrt{2\pi(2g-1)}};$$

$$1/\binom{n}{4g} > \frac{(4g)^{4g}\sqrt{2\pi\cdot 4g}\exp[-4g-\frac{1}{360(4g)^3}+\frac{1}{12(4g)}]}{n^{4g}}.$$

Multiplying these terms together, we have:

$$\Pr_{4g}[d=2] > \tfrac{\kappa_3(n)}{3\sqrt{2\pi\log n}\cdot n^{4-\log 9}},$$

where $\kappa_3(n)$ is as follows and one could verify that $\kappa_3(n) = 1-o(1)$:

$$\kappa_3(n) = \frac{(1+\frac{4(\sqrt{n}-2g-1)}{n})^{2g+1}(1-\frac{4(\sqrt{n}+2g-1)}{3n})^{2g-1}}{(1+\frac{1}{2g})^{2g+1}(1-\frac{1}{2g})^{2g-1}\sqrt{1-\frac{1}{4g^2}}}$$
$$\cdot\exp\left[\frac{1}{12(4g)}-\frac{1}{12(2g+1)}-\frac{1}{12(2g-1)}-\frac{1}{360(4g)^3}\right].$$

By Lemma 3, we know that the probability for RLS_{4g} to make any improvement reaches its minimum at $|x|_1 = \frac{3n}{4} - \sqrt{n}$ during the current stage. Thus, by using the same strategy as before, we have:

$$\Pr[\overline{\mathcal{A}}_{4g}] < 1 - \left(1 - \left(1 - \frac{\kappa_3(n)}{3\sqrt{2\pi \log n} n^{4-\log 9}}\right)^{\tau}\right)^{n/8-\sqrt{n}/2} < \frac{n}{\exp[n^2]}.$$

Similarly, $\mathbb{E}[T_1 \mid \mathrm{RLS}_{4g}, \overline{\mathcal{A}}_{4g}]$ can be bounded by $\mathcal{O}(n\tau)$. And if $\mathcal{A}_{4g}$ occurs, the algorithm will at most improve by 2 for $n/8 - \sqrt{n}/2$ times, and the expected time for each improvement can be bounded by that at $|x|_1 = \frac{3n}{4} - \sqrt{n}$ which is $\mathcal{O}(n^{4-\log 9}\sqrt{\log n})$. Thus,

$$\mathbb{E}[T_1 \mid \mathrm{RLS}_{4g}, \mathcal{A}_{4g}] < \frac{n}{8}\mathcal{O}(n^{4-\log 9}\sqrt{\log n}) = \mathcal{O}(n^{5-\log 9}\sqrt{\log n}).$$

Hence, we can get:

$$\begin{aligned}\mathbb{E}[T_1 \mid \mathrm{RLS}_{4g}] &< \mathcal{O}(n^{5-\log 9}\sqrt{\log n}) + \mathcal{O}(n\tau)\frac{n}{\exp[n^2]} \\ &= \mathcal{O}(n^{5-\log 9}\sqrt{\log n}).\end{aligned}$$

Overall, we have that:

$$\mathbb{E}[T_1] = \frac{1}{2}\mathcal{O}(n \log n) + \frac{1}{2}\mathcal{O}(n^{5-\log 9}\sqrt{\log n}) = \mathcal{O}(n^{5-\log 9}\sqrt{\log n}).$$

Now, we start to bound $\mathbb{E}[T_2]$. Note that once RLS_1 is chosen, it will directly fail after τ iterations, thus

$$\mathbb{E}[T_2] = \frac{1}{2}(\tau + \mathbb{E}[T_2]) + \frac{1}{2}\mathbb{E}[T_2 \mid \mathrm{RLS}_{4g}] = \tau + \mathbb{E}[T_2 \mid \mathrm{RLS}_{4g}].$$

Denote by ${}_i\Pr_j$ the probability of using RLS_i to jump over a hurdle of length j. We know from Theorem 1 that:

$${}_{4g}\Pr{}_{2g} = (1 - o(1))\sqrt{2/(3\pi \log n)}n^{-\log 9};$$

and,

$${}_{4g}\Pr{}_{g} = (1 - o(1))\sqrt{8/(15\pi \log n)}n^{-\log(25\sqrt{5}/16)}.$$

Denote by $\mathcal{E}$ the event that once RLS_{4g} is chosen, it never fails during the current stage. Without loss of generality, one can assume that the number of $2g$ hurdles and g hurdles are equivalent. Then,

$$\Pr[\overline{\mathcal{E}}] = 1 - [(1 - (1 - {}_{4g}\Pr{}_{2g})^{\tau})(1 - (1 - {}_{4g}\Pr{}_{g})^{\tau})]^{\sqrt{n}/(3\log n)}.$$

Since we have ${}_{4g}\Pr_{2g}¿{}_{4g}\Pr_g$ and $\tau \geq n^{\log 9+\epsilon}$, we can bound $\Pr[\overline{\mathcal{E}}]$ by:

$$\Pr[\overline{\mathcal{E}}] < 1 - \left(1 - \left(1 - \frac{(1-o(1))\sqrt{2}}{n^{\log 9}\sqrt{3\pi \log n}}\right)^{n^{\log 9+\epsilon}}\right)^{\frac{2\sqrt{n}}{3\log n}}.$$

Here we again use the inequality $(1 - 1/x)^x < 1/\mathrm{e} < (1 - 1/x)^{x-1}$ and $1 - \exp[-x] \leq x$ to get:

$$\Pr[\overline{\mathcal{E}}] < \frac{\sqrt{n}}{\log n \left(\exp\left[\frac{(1-o(1))n^{\epsilon}}{\sqrt{3\pi \log n}}\right] - 1\right)}.$$

If $\overline{\mathcal{E}}$ happens, then in the worst case, RLS_{4g} will directly fail after each improvement. Denote the runtime for such a process by $\tilde{T}$. Now we start to bound the expected runtime $\mathbb{E}[\tilde{T} \mid \text{OM} = \frac{3n}{4} - \sqrt{n}]$. Using the law of total expectation, we can write:

$$\begin{aligned}\mathbb{E}[\tilde{T} \mid \text{OM} = \frac{3n}{4} - \sqrt{n}] = \frac{1}{2}\mathbb{E}\left[\tilde{T} \mid \text{OM} = \frac{3n}{4} - \sqrt{n}, \text{RLS}_1\right] \\ + \frac{1}{2}\mathbb{E}\left[\tilde{T} \mid \text{OM} = \frac{3n}{4} - \sqrt{n}, \text{RLS}_{4g}\right].\end{aligned}$$

Thus, we have $\mathbb{E}[\tilde{T} \mid \text{OM} = \frac{3n}{4} - \sqrt{n}] = \tau + \mathbb{E}[\tilde{T} \mid \text{OM} = \frac{3n}{4} - \sqrt{n}, \text{RLS}_{4g}]$. Once we choose RLS_{4g}, it will either improve within τ iterations or fail after τ iterations. Conditioning on the event that RLS_{4g} makes an improvement within τ steps, we have that the expected time it spends on making an improvement is:

$$\frac{1}{{}_{4g}\Pr{}_{2g}} - \frac{(1 - {}_{4g}\Pr{}_{2g})^{\tau}\tau}{1 - (1 - {}_{4g}\Pr{}_{2g})^{\tau}} < \frac{1}{{}_{4g}\Pr_{2g}} < (1 + o(1))\sqrt{3\pi \log n/2} \cdot n^{\log 9}.$$

Similarly, we can obtain that the expected runtime of RLS_{4g} to cross a g hurdle is smaller than $(1 + o(1))\sqrt{15\pi \log n/8}n^{\log(25\sqrt{5}/16)}$.

The probability for the event that RLS_{4g} fails after τ iterations, denoted by $\overline{E}$, is smaller than $(1 - \frac{(1-o(1))\sqrt{2}}{n^{\log 9}\sqrt{3\pi \log n}})^{n^{\log 9+\epsilon}}$ which is further smaller than $\exp[-\frac{(1-o(1))\sqrt{2}n^{\epsilon}}{\sqrt{3\pi \log n}}] = o(1)$. Thus we have:

$$\begin{aligned}&\mathbb{E}\left[\tilde{T} \mid \text{OM} = \frac{3n}{4} - \sqrt{n}, \text{RLS}_{4g}\right] \\ &< \Pr[E]\left((1 + o(1))\sqrt{3\pi \log n/2} \cdot n^{\log 9} + \tau + \mathbb{E}\left[\tilde{T} \mid \text{OM} = \frac{3n}{4} - \sqrt{n} + 2\log n\right]\right) \\ &+ \Pr[\overline{E}]\left(\tau + \mathbb{E}\left[\tilde{T} \mid \text{OM} = \frac{3n}{4} - \sqrt{n}\right]\right).\end{aligned}$$

Thus,

$$\mathbb{E}\left[\tilde{T} \mid \text{OM} = \frac{3n}{4} - \sqrt{n}\right] < \frac{\tau + \Pr[\overline{E}]\tau}{\Pr[E]} + \mathcal{O}(\tau) + \mathbb{E}\left[\tilde{T} \mid \text{OM} = \frac{3n}{4} - \sqrt{n} + 2\log n\right].$$

We see from above that under our assumption that RLS_{4g} will directly fail after each improvement, such a process will cost $\mathcal{O}(\tau)$ iterations to jump the first hurdle and start jumping the second without any conditioning. As a whole, we can bound $\mathbb{E}[\tilde{T} \mid \text{OM} = \frac{3n}{4} - \sqrt{n}]$ by $\frac{2\sqrt{n}}{3\log n}\mathcal{O}(\tau)$. Thus $\mathbb{E}[T_2 \mid \text{RLS}_{4g}, \overline{\mathcal{E}}] = \mathcal{O}(\frac{\sqrt{n}\tau}{\log n})$.

Otherwise, if $\mathcal{E}$ occurs, then the expected runtime for each improvement, conditioning on RLS_{4g} succeeding within τ iterations, is $\mathcal{O}(n^{\log 9}\sqrt{\log n})$. From our calculation above, $\frac{2\sqrt{n}}{3\log n}$ such improvements are required to hit the optima of CUTTWORATES.

Combining them together, we have that

$$\mathbb{E}[T_2] < \tau + \Pr[\mathcal{E}]\mathcal{O}\left(\frac{n^{\log 9}\sqrt{n}}{\sqrt{\log n}}\right) + \Pr[\overline{\mathcal{E}}]\mathcal{O}\left(\frac{\sqrt{n}\tau}{\log n}\right).$$

Since $\Pr[\mathcal{E}] < 1$ and $\Pr[\overline{\mathcal{E}}]\mathcal{O}(\sqrt{n}\tau/\log n) = o(1)$, we have

$$\mathbb{E}[T_2] = \mathcal{O}\left(\tau + \frac{n^{\log 9}\sqrt{n}}{\sqrt{\log n}}\right).$$

To bound $\mathbb{E}[S]$, we know that only RLS_{4g} can improve without failure from stage 1 to stage 2, and it will take an expected number of $\mathcal{O}(\frac{n^{\log 9}\sqrt{n}}{\sqrt{\log n}})$ iterations before it fails within stage 2. Thus $\mathbb{E}[S] = \mathcal{O}(\frac{n^{\log 9}\sqrt{n}}{\sqrt{\log n}})$. Overall, we have:

$$\mathbb{E}[T] = \mathcal{O}(n^{5-\log 9}\sqrt{\log n}) + \mathcal{O}\left(\tau + \frac{n^{\log 9}\sqrt{n}}{\sqrt{\log n}}\right) + \mathcal{O}\left(\frac{n^{\log 9}\sqrt{n}}{\sqrt{\log n}}\right),$$

which is $\mathcal{O}(\tau + n^{\log 9}\sqrt{n/\log n})$. □

We now consider the complete TWORATES function.

Theorem 4. *The expected runtime of the Generalised Random Gradient Hyper-heuristic with $\tau \geq n^{\log 9+\epsilon}$ and $\tau = poly(n)$, equipped with $H = \{\text{RLS}_1, \text{RLS}_{4g}\}$ to optimise* TWORATES *is $\mathcal{O}(n\tau)$.*

Proof. Like for the proof of Theorem 3, we bound the expected runtime by $\mathbb{E}[T] < \mathbb{E}[T_1] + \mathbb{E}[T_2] + \mathbb{E}[S]$ where $\mathbb{E}[T_1]$ and $\mathbb{E}[T_2]$ denote the expected runtime on the CUTTWORATES part and the second ONEMAX part, while $\mathbb{E}[S]$ denotes the expected number of iterations between the first solution being constructed and the first operator choice occuring in the second ONEMAX part. We know from Theorem 3 and Lemma 4 that $\mathbb{E}[T_1] = \mathcal{O}(\tau + n^{\log 9}\sqrt{n/\log n})$, and $\mathbb{E}[T_2] = \mathcal{O}(n\tau)$. Since each improvement takes at most τ iterations and at most $\frac{n}{4}$ improvements can be made during the second ONEMAX part we get $\mathbb{E}[S] = \mathcal{O}(n\tau)$. Hence, we conclude that the overall expected runtime is $\mathcal{O}(n\tau)$. □

While the proof of the theorem uses a simple argument to bound the expected time on the final ONEMAX slope, the empirical analysis provided in the experiments section suggests that the bound may not be very loose.

5. Using the Complete Heuristic Set

In the previous section no asymptotic speed ups could be derived for GRG repeatedly applying the optimal operator across the HURDLE region of the function over random selection when the two optimal operators are used. In this section we show that if the complete heuristic set is employed, then GRG is considerably faster than using random selection (i.e., the HH learns which operator to use). Our main result in this section is the following.

Theorem 5. *The expected runtime of the Generalised Random Gradient Hyper-heuristic with $\tau \geq n^{\log 9+\epsilon}$, $\tau = poly(n)$, equipped with $H = \{RLS_1 \ldots, RLS_n\}$ for* TWORATES *is $\mathcal{O}(n\tau \log n)$.*

We will bound the expected times for the final ONEMAX region and the HURDLE region of the function in separate lemmata, and then put the results together to prove the theorem. We conclude the section by providing a lower bound on the runtime of the Simple Random HH to complete our claim.

We begin with the second ONEMAX part.

Lemma 5. *The expected runtime of the Generalised Random Gradient Hyper-heuristic equipped with $H := \{\text{RLS}_1, \ldots, \text{RLS}_n\}$ and $\tau \geq n^{\log 9+\epsilon}$, $\tau = poly(n)$, starting from a random heuristic choice at $|x|_1 \geq 3n/4$ on* TWORATES *is $\mathcal{O}(n\tau \log n)$.*

Proof. Throughout this proof, we pessimistically assume the setting that once an operator makes any improvement, the fitness improves only by 1 and the number of iterations required to make the improvement is exactly τ. This process provides an upper bound on the expected time for GRG to optimise the second ONEMAX part of the TWORATES function.

Let $\mathbb{E}[T \mid \text{OM} = k]$ denote the expected time required for GRG to solve ONEMAX when the current solution has a fitness of k in the described setting, and let $\mathbb{E}[T \mid \text{OM} = k, \text{RLS}_i]$ denote the expected runtime starting from a solution with k 1-bits to solve ONEMAX using RLS_i in the described setting. We are seeking an upper bound for $\mathbb{E}[T \mid \text{OM} = 3n/4] \leq \mathbb{E}[T \mid \text{OM} = k^*]$, where

$$k^* = \arg \max_{k \in [3n/4..n]} \mathbb{E}[T \mid \text{OM} = k].$$

We partition the heuristics into three sets, H_1, H_2 and H_3 (where $H_1 \cup H_2 \cup H_3 = H$):

- H_1 contains only RLS_1, which is the best operator for the second ONEMAX part of TWORATES;
- H_3 contains the operators that have a probability to find an improvement in the second ONE-MAX part that is always smaller that $1/n^2$ (i.e. if p is the probability that the current operator will make an improvement in a single step, then $1-(1-p)^\tau < 1/n^2$);
- H_2 contains the remaining operators.

By the law of total expectation, we have,

$$\begin{aligned}\mathbb{E}[T \mid \text{OM} = k^*] = \frac{1}{n} \cdot \mathbb{E}[T \mid \text{OM} = k^*, \text{RLS}_1] + \frac{1}{n} \cdot \sum_{i \in H_2} \mathbb{E}[T \mid \text{OM} = k^*, \text{RLS}_i] \\ + \frac{1}{n} \cdot \sum_{i \in H_3} \mathbb{E}[T \mid \text{OM} = k^*, \text{RLS}_i]. \qquad (1)\end{aligned}$$

We now seek a bound on each of these summands.

Denote by G_i the event that once the operator RLS_i is chosen, it never fails until hitting the optimum. We can therefore write

$$\begin{aligned}\mathbb{E}[T \mid \text{OM} = k^*, \text{RLS}_i] = \Pr[G_i] \cdot \mathbb{E}[T \mid \text{OM} = k^*, \text{RLS}_i, G_i] \\ + \Pr[\overline{G}_i] \cdot \mathbb{E}[T \mid \text{OM} = k^*, \text{RLS}_i, \overline{G}_i],\end{aligned}$$

where for $i = 1$, we have

$$\Pr[G_1] = \prod_{i=1}^{n-k^*} [1-(1-i/n)^\tau] > \prod_{i=1}^{n/4} [1-(1-i/n)^\tau] > [1-(1-1/n)^\tau]^{n/4} = 1 - \exp[-\Omega(n)].$$

By Lemma 4, we can bound $\mathbb{E}[T \mid \text{OM} = k^*, \text{RLS}_1, \overline{G}_1] = \mathcal{O}(n^2\tau)$ since $|H| = n$. Thus, $\Pr[\overline{G}_1] \cdot \mathbb{E}[T \mid \text{OM} = k^*, \text{RLS}_1, \overline{G}_1] = \exp[-\Omega(n)] \cdot \mathcal{O}(n^2\tau) = o(1)$ since $\tau = \mathit{poly}(n)$. Further, we bound $\Pr[G_1] \cdot \mathbb{E}[T \mid \text{OM} = k^*, \text{RLS}_1, G_1] < 1 \cdot \frac{n\tau}{4}$. Overall, we have

$$\mathbb{E}[T \mid \text{OM} = k^*, \text{RLS}_1] = \mathcal{O}(n\tau).$$

We now bound the expected runtime for the operators in H_2. We firstly note that $\Pr[G_i] \cdot \mathbb{E}[T \mid \text{OM} = k^*, \text{RLS}_i, G_i] \leq (n - k^*)\tau \leq \frac{n\tau}{4} = \mathcal{O}(n\tau)$ holds for $\text{RLS}_i \in H_2$. If $\overline{G}_i$ occurs, since we do not precisely know which OneMax value RLS_i will fail at, we let θ denote the number of improvements made by RLS_i consecutively before failing. Then, the expected runtime given that $\overline{G}_i$ occurs can be written as,

$$\mathbb{E}[T \mid \text{OM} = k^*, \text{RLS}_i, \overline{G}_i] = \tau\theta + \tau + \mathbb{E}[T \mid \text{OM} = k^* + \theta],$$

where $\theta \in [0..n - k^*]$. Since $\theta \leq n - k^* \leq \frac{n}{4}$ and $\mathbb{E}[T \mid \text{OM} = k^* + \theta] < \mathbb{E}[T \mid \text{OM} = k^*]$ (by the definition of k^*), for any $i \in H_2$, we have:

$$\mathbb{E}[T \mid \text{OM} = k^*, \text{RLS}_i] < \mathcal{O}(n\tau) + \mathbb{E}[T \mid \text{OM} = k^*].$$

We now derive bounds for the operators in H_3. Denote by $\mathcal{G}_i$ the event that the operator RLS_i, once chosen, does not fail to make its first improvement. Recall that for all operators in H_3, $\mathcal{G}_i < 1/n^2$. We have that

$$\begin{aligned}\mathbb{E}[T \mid \text{OM} = k^*, \text{RLS}_i] = {} & \Pr[\mathcal{G}_i] \cdot \mathbb{E}[T \mid \text{OM} = k^*, \text{RLS}_i, \mathcal{G}_i] \\ & + \Pr[\overline{\mathcal{G}}_i] \cdot \mathbb{E}[T \mid \text{OM} = k^*, \text{RLS}_i, \overline{\mathcal{G}}_i].\end{aligned}$$

Again, the expected runtime given $\mathcal{G}_i$ is $\mathcal{O}(n^2\tau)$ by Lemma 4. Otherwise, if $\overline{\mathcal{G}}_i$ occurs, then the expected runtime can be expressed as $\tau + \mathbb{E}[T \mid \text{OM} = k^*]$ (i.e., the algorithm will fail after τ iterations and re-select operators again). Therefore, we can bound the expected runtime to find the optimum, given that $\text{RLS}_{i \in H_3}$ is chosen first from above by

$$\begin{aligned}\mathbb{E}[T \mid \text{OM} = k^*, \text{RLS}_{i \in H_3}] & \leq 1/n^2 \cdot \mathcal{O}(n^2\tau) + \tau + \mathbb{E}[T \mid \text{OM} = k^*] \\ & = \mathcal{O}(\tau) + \mathbb{E}[T \mid \text{OM} = k^*].\end{aligned}$$

Finally, we seek a bound on the size of the set H_3 (i.e., $|H_3|$). For a given operator RLS_m, we know by Lemma 1 that the probability for it to make any improvement in a single step at $\text{OM} = a \geq 3n/4$ has the following bound:

$$\Pr_m[d \geq 0] \leq \exp\left[-2\left(\frac{a}{n} - \frac{1}{2}\right)^2 m\right] \leq \exp\left[-m/8\right].$$

If $m \geq 8\ln(n^2\tau + 1)$, then

$$1 - (1 - \Pr_m[d \geq 0])^\tau \leq 1 - \left(1 - \frac{1}{n^2\tau + 1}\right)^\tau \leq 1 - \exp\left[-\frac{1}{n^2}\right] \leq \frac{1}{n^2}.$$

Therefore $|H_3| > n - 8\ln(n^2\tau + 1)$. Since $\tau = \mathit{poly}(n)$, $|H_3| = n - \mathcal{O}(\log n)$. We further know that $|H_2| = n - |H_1| - |H_3| < n - 1 - (n - 8\ln(n^2\tau + 1)) = \mathcal{O}(\log n)$.

We now return to Equation 1 and input the given bounds:

$$\begin{aligned}
&\mathbb{E}[T \mid \text{OM} = k^*] \\
&\leq \frac{\mathcal{O}(n\tau)}{n} + \frac{(n - |H_3| - 1)}{n} \cdot (\mathcal{O}(n\tau) + \mathbb{E}[T \mid \text{OM} = k^*]) + \frac{|H_3|}{n} \cdot (\mathcal{O}(\tau) + \mathbb{E}[T \mid \text{OM} = k^*]) \\
&= \frac{n-1}{n} \cdot \mathbb{E}[T \mid \text{OM} = k^*]) + \frac{\mathcal{O}(n\tau)}{n} + \frac{(n - |H_3| - 1)}{n} \cdot \mathcal{O}(n\tau) + \frac{|H_3|}{n} \cdot \mathcal{O}(\tau).
\end{aligned}$$

Therefore,

$$\begin{aligned}
\mathbb{E}[T \mid \text{OM} = k^*] &\leq \mathcal{O}(n\tau) + (n - |H_3| - 1) \cdot \mathcal{O}(n\tau) + |H_3| \cdot \mathcal{O}(\tau) \\
&\leq \mathcal{O}(n\tau) + \mathcal{O}(n\tau \log n) + \mathcal{O}(n\tau) \\
&= \mathcal{O}(n\tau \log n).
\end{aligned}$$

This concludes our proof. □

We now bound the expected time for GRG to cross the HURDLE region.

Lemma 6. *Starting with a random heuristic choice at the beginning of any hurdle in the* HURDLE *region, the expected runtime of the Generalised Random Gradient Hyper-heuristic with* $\tau \geq n^{\log 9+\epsilon}$, $\tau = poly(n)$, *equipped with* $H = \{RLS_1 \dots RLS_n\}$, *to optimise* CUTTWORATES *is* $\mathcal{O}(n\tau)$.

Proof. As with the proof of Lemma 5, we again divide the operators into three set: H_1, H_2 and H_3. H_1 contains only the operator RLS_{4g}, which is the best operator for the HURDLE region (see Section 3); H_2 contains all operators that are able to make some progress in the HURDLE region with high probability, but not all consecutively; H_3 contains those that will directly fail to find an improvement within τ iterations when selected. We note that $|H_3| = n - \Omega(\log n)$ since any RLS_k with $k = \omega(\log n)$ has super-polynomially small probability to make any improvement, and there is 0 probability of improvement when $1 \leq k \leq g - 1$. Denote $|H_3| = n - \hat{c} \log n$ for some constant $\hat{c} > 0$, and let T denote the runtime to find the global optimum of CUTTWORATES when starting with a random operator selection.

We call a number λ *feasible* if the TWORATES value at $\text{OM} = \lambda$ is greater than zero. Then we let,

$$s^* = \arg \max_{\substack{\lambda \textit{ feasible}, \\ \lambda \in [s, 3n/4]}} \{\mathbb{E}[T \mid \text{OM} = \lambda]\}.$$

We can write $\mathbb{E}[T \mid \text{OM} = s^*]$ as,

$$\begin{aligned}
\mathbb{E}[T \mid \text{OM} = s^*] = &\frac{1}{n}\mathbb{E}[T \mid \text{OM} = s^*, \text{RLS}_{4g}] \\
&+ \frac{1}{n} \sum_{\text{RLS}_i \in H_2} \mathbb{E}[T \mid \text{OM} = s^*, \text{RLS}_i] \\
&+ \frac{1}{n} \sum_{\text{RLS}_i \in H_3} \mathbb{E}[T \mid \text{OM} = s^*, \text{RLS}_i].
\end{aligned}$$

We now provide a bound on the probability of unlikely events during this process (i.e., if RLS_{4g} cannot make consecutive improvements, or if an operator in H_3 makes an improvement). To bound the expected runtime in these cases, we pessimistically assume that for each operator, every improvement takes time τ and improves only by either $2g$ or g (dependent on the hurdle jump size); we further assume it will directly fail in the τ iterations after any success. We model this via a Markov chain with absorbing time that forms an upper bound on the expected runtime of GRG taking into account these unlikely events.

Denote by $T^\dagger$ the runtime for such a Markov chain to make the first jump and then fail with the successful operator. Let P_F denote the probability that RLS_{4g} fails to find an improvement within τ steps. With probability $1 - P_F$, RLS_{4g} makes an improvement of size $2g$ within τ steps. Otherwise, with probability P_F, RLS_{4g} fails, and we wait for RLS_{4g} to be chosen again which happens in expected time $\tau(n-1)$, where we are pessimistically assuming any other operator will fail directly. Then, we have:

$$\mathbb{E}[T^\dagger] \le \frac{1}{n}\left(P_F\left(\tau + \mathbb{E}[T^\dagger]\right) + (1-P_F)2\tau\right) + \frac{n-1}{n}\left(\tau + \mathbb{E}[T^\dagger]\right).$$

So,

$$\mathbb{E}[T^\dagger] \le 2\tau + \frac{n-1+P_F}{1-P_F}\cdot\tau < 2\tau + \frac{n}{1-P_F}\tau.$$

We know that when $\tau \ge n^{\log 9+\epsilon}$, $P_F < \exp[-\frac{(1-o(1))\sqrt{2}n^\epsilon}{\sqrt{3\pi\log n}}] = \exp[-\Omega(n^\epsilon/\sqrt{\log n})]$. Therefore, the expected time of this setting to reach the optimum of CUTTWORATES is bounded by

$$\frac{2\sqrt{n}}{3\log n}\left(2\tau + \frac{n\tau}{1-o(1)}\right) = \mathcal{O}(\tau n\sqrt{n}/\log n) = poly(n).$$

We already know from the proof of Theorem 3 that the probability for RLS_{4g} to solve all hurdles consecutively until hitting the optima at $\mathrm{OM} = \frac{3n}{4}$ is greater than $1 - \frac{\sqrt{n}}{\log n(\exp[\frac{(1-o(1))n^\epsilon}{\sqrt{3\pi\log n}}]-1)}$.

Thus,

$$\mathbb{E}[T \mid \mathrm{OM} = s^*, \mathrm{RLS}_{4g}] < \frac{2\sqrt{n}\tau}{3\log n} + \frac{\sqrt{n}\cdot\mathcal{O}\left(\frac{n\tau\sqrt{n}}{\log n}\right)}{\log n\left(\exp\left[\frac{(1-o(1))n^\epsilon}{\sqrt{3\pi\log n}}\right]-1\right)},$$

which is $\mathcal{O}\left(\frac{\sqrt{n}\tau}{\log n}\right) + o(1)$. Similarly, we can bound the expected runtime for operators in H_3 by

$$\mathbb{E}[T \mid \mathrm{OM} = s^*, \mathrm{RLS}_i \in H_3] < \tau + \mathbb{E}[T \mid \mathrm{OM} = s] + o(1).$$

For H_2, the operators may improve l times and then fail at $\mathrm{OM} = \lambda \ge s$ where λ is feasible. This can be expressed as:

$$\mathbb{E}[T \mid \mathrm{OM} = s^*, \mathrm{RLS}_i \in H_2] = l\tau + \tau + \mathbb{E}\left[T \mid \mathrm{OM} = \lambda\right].$$

Note that $l \le \frac{2\sqrt{n}}{3\log n}$ and $\mathbb{E}[T \mid \mathrm{OM} = \lambda] \le \mathbb{E}[T \mid \mathrm{OM} = s^*]$. Thus,

$$\mathbb{E}[T \mid \mathrm{OM} = s^*, \mathrm{RLS}_i \in H_2] \le \frac{2\sqrt{n}\tau}{3\log n} + \tau + \mathbb{E}[T \mid \mathrm{OM} = s^*].$$

Thus, we have

$$\begin{aligned}\mathbb{E}[T \mid \text{OM} = s^*] \le \frac{1}{n}\Big(&\mathcal{O}\left(\frac{\sqrt{n}\tau}{\log n}\right)\\ &+ (\hat{c}\log n - 1)\left(\mathcal{O}\left(\frac{\sqrt{n}\tau}{\log n}\right) + \mathbb{E}[T \mid \text{OM} = s^*]\right)\\ &+ (n - \hat{c}\log n)(\mathcal{O}(\tau) + \mathbb{E}[T \mid \text{OM} = s^*])\Big),\end{aligned}$$

which gives us $\mathbb{E}[T \mid \text{OM} = s^*] = \mathcal{O}(n\tau)$ which concludes our proof. □

Now we combine Lemma 5 and Lemma 6 to prove the main result of the section.

Proof (of Theorem 5). We seperate the runtime of the Generalised Random Gradient Hyper-heuristic on the whole TwoRates function into three stages: the first OneMax stage (i.e., when the current TwoRates value is less than or equal to $3n/4 - \sqrt{n}$), the Hurdle stage (i.e., when current TwoRates value is between $3n/4 - \sqrt{n}$ and $3n/4$), and the second OneMax stage (i.e., when the current TwoRates value is greater than $3n/4$). Let T_1, T_2 and T_3 respectively denote the time between the first random operator choice in each of these stages of the optimisation process and the moment when the necessary fitness value to enter the subsequent stage is reached (or the global optimum is found). Further, let S_2 (and S_3) denote the expected number of iterations between the first solution in stage 2 (stage 3) being constructed and the first operator choice occurring in stage 2 (stage 3). Then, for the runtime T of GRG on TwoRates, $\mathbb{E}[T]$ can be calculated by

$$\mathbb{E}[T] < \mathbb{E}[T_1] + \mathbb{E}[T_2] + \mathbb{E}[T_3] + \mathbb{E}[S_2] + \mathbb{E}[S_3].$$

By Lemma 5 , we know that $\mathbb{E}[T_3] = \mathcal{O}(n\tau\log n)$, which also bounds $\mathbb{E}[T_1]$, and by Lemma 6, we know that $\mathbb{E}[T_2] = \mathcal{O}(n\tau)$. We know that an operator will make at most $\frac{2\sqrt{n}}{3\log n}$ improvements before failure during the Hurdle stage, and each improvement takes at most τ iterations, so $\mathbb{E}[S_2]$ can be bounded by $\frac{2\sqrt{n}\tau}{3\log n}$. Similarly, during the second OneMax stage, one operator will make at most $\frac{n}{4}$ improvements, thus $\mathbb{E}[S_3]$ can be bounded by $\frac{n\tau}{4}$. Combining all the inequalities above we conclude that $\mathbb{E}[T] = \mathcal{O}(n\tau\log n)$. □

Finally, we show that the Simple Random HH is slower.

Theorem 6. *The expected runtime for the Simple Random Hyper-heuristic to optimise* TwoRates*, equipped with $H = \{RLS_1 \dots RLS_n\}$, is $\Omega\left(\frac{n^{3/2+\log 9}}{\log n\sqrt{\log n}}\right)$.*

Proof. We bound the expected runtime of the Simple Random SHH on TwoRates from below by the expected time for it to cross the Hurdle region.

We firstly note that while RLS$_{4g}$ is the best operator to jump the $2g$ hurdle, the probability of any operator making an improvement of more than $2g$ (i.e., RLS$_{6g}$ jumping two hurdles at once and making an improvement of $3g$) is at least $1/n$ smaller than the probability of RLS$_{4g}$ to make an improvement of $2g$, which is $\frac{(1-o(1))\sqrt{2}}{\sqrt{3\pi\log n}n^{\log 9}}$ by Theorem 1. Moreover, we know from Lemma 1 that the probability for RLS$_k$ to make any improvement in the Hurdle region for $k \ge 30\log n$ is smaller than $1/n^5$.

We separate the operators into two sets: the first contains the operators RLS_m where $2\log n \leq m \leq 30\log n - 1$, and the second set contains the remaining operators. Within the second set, we know that the probability to jump the first hurdle is at most $1/n^5$. We assume optimistically for the first set that all operators have the same probability as RLS_{4g} to jump the first hurdle. Let T denote the time to jump the first hurdle of size $2g$ for this process. Then, the following holds:

$$\begin{aligned}\mathbb{E}[T] \geq \frac{28\log n}{n}\left(\frac{(1-o(1))\sqrt{2}}{\sqrt{3\pi\log n}n^{\log 9}} + \left(1 - \frac{(1-o(1))\sqrt{2}}{\sqrt{3\pi\log n}n^{\log 9}}\right)(1+\mathbb{E}[T])\right) \\ + \frac{n-28\log n}{n}\left(1-\frac{1}{n^5}\right)(1+\mathbb{E}[T]).\end{aligned}$$

We define $g(n)$ to simplify the right hand side

$$g(n) := \frac{(1-o(1))\sqrt{2}}{\sqrt{3\pi\log n}n^{\log 9}}.$$

Thus we have

$$\mathbb{E}[T] \geq \frac{28\log n}{n}\Big(1+(1-g(n))\mathbb{E}[T]\Big) + \frac{n-28\log n}{n}\left(1-\frac{1}{n^5}\right)(1+\mathbb{E}[T]). \tag{2}$$

Let

$$A := \frac{28\log n}{n}\Big(1+(1-g(n))\mathbb{E}[T]\Big),$$

and

$$B := \frac{n-28\log n}{n}\left(1-\frac{1}{n^5}\right).$$

Then we can rewrite the inequality (2) as

$$\mathbb{E}[T] \geq A + B(1+\mathbb{E}[T]) = A + B + B\cdot\mathbb{E}[T].$$

Also, we have:

$$B = 1 - \frac{28\log n}{n} - \frac{1}{n^5} + \frac{28\log n}{n^6} \geq 1 - \frac{28\log n}{n} - \frac{1}{n^5}.$$

Hence,

$$\mathbb{E}[T] \geq A + 1 - \frac{28\log n}{n} - \frac{1}{n^5} + \left(1 - \frac{28\log n}{n} - \frac{1}{n^5}\right)\mathbb{E}[T].$$

After moving the third term to the left hand side and dividing both sides of the inequality by $28\log n/n + 1/n^5$, we have that

$$\mathbb{E}[T] \geq \frac{A}{\frac{28\log n}{n}+\frac{1}{n^5}} + \frac{1-\frac{28\log n}{n}-\frac{1}{n^5}}{\frac{28\log n}{n}+\frac{1}{n^5}}.$$

The first term is:

$$\frac{A}{\frac{28\log n}{n}+\frac{1}{n^5}} = \frac{\frac{28\log n}{n}\Big(1+(1-g(n))\mathbb{E}[T]\Big)}{\frac{2logn}{n}+\frac{1}{n^5}} = \frac{1}{1+\frac{1}{28n^4\log n}}\Big(1+(1-g(n))\mathbb{E}[T]\Big).$$

Here we use the inequality $\frac{1}{1+x} > 1 - x$ for $x > 0$, and obtain that

$$\begin{aligned}\frac{A}{\frac{28\log n}{n} + \frac{1}{n^5}} &> \left(1 - \frac{1}{28n^4 \log n}\right)\Big(1 + (1-g(n))\mathbb{E}[T]\Big)\\ &= 1 - \frac{1}{28n^4\log n} + \left(1 - \frac{1}{28n^4\log n}\right)(1-g(n))\mathbb{E}[T].\end{aligned}$$

We know

$$1 - \frac{1}{28n^4\log n} > 0,$$

and

$$\left(1 - \frac{1}{28n^4\log n}\right)(1-g(n))\mathbb{E}[T] > \left(1 - \frac{1}{28n^4\log n} - g(n)\right)\mathbb{E}[T].$$

Thus

$$\frac{A}{\frac{28\log n}{n} + \frac{1}{n^5}} > \left(1 - \frac{1}{28n^4\log n} - g(n)\right)\mathbb{E}[T].$$

As for the second term, we can rewrite it as:

$$\frac{1 - \frac{28\log n}{n} - \frac{1}{n^5}}{\frac{28\log n}{n} + \frac{1}{n^5}} = \frac{n}{28\log n}\frac{1 - \frac{28\log n}{n} - \frac{1}{n^5}}{1 + \frac{1}{28n^4\log n}}.$$

Eventually we have

$$\mathbb{E}[T] > \frac{n}{28\log n}\frac{1 - \frac{28\log n}{n} - \frac{1}{n^5}}{1 + \frac{1}{28n^4\log n}} + \left(1 - \frac{1}{28n^4\log n} - g(n)\right)\mathbb{E}[T].$$

After moving the second term to the left hand side and dividing both sides of the inequality by $\frac{1}{28n^4\log n} + g(n)$, we can obtain that

$$\mathbb{E}[T] > \frac{\frac{n}{28\log n}\frac{1 - \frac{28\log n}{n} - \frac{1}{n^5}}{1 + \frac{1}{28n^4\log n}}}{\frac{1}{28n^4\log n} + g(n)}.$$

As for the denominator, we substitute $g(n)$ back:

$$\frac{1}{28n^4\log n} + g(n) = \frac{1}{28n^4\log n} + \frac{(1-o(1))\sqrt{2}}{\sqrt{3\pi\log n}n^{\log 9}},$$

which equals to

$$\frac{(1-o(1))\sqrt{2}}{\sqrt{3\pi\log n}n^{\log 9}}\left(1 + \frac{\sqrt{3\pi}}{(1-o(1))28\sqrt{2}n^{4-\log 9}\sqrt{\log n}}\right).$$

Finally, we have,

$$\begin{aligned}\mathbb{E}[T] &> \frac{\frac{n}{28\log n}\frac{1 - \frac{28\log n}{n} - \frac{1}{n^5}}{1 + \frac{1}{28n^4\log n}}}{\frac{(1-o(1))\sqrt{2}}{\sqrt{3\pi\log n}n^{\log 9}}\left(1 + \frac{\sqrt{3\pi}}{(1-o(1))28\sqrt{2}n^{4-\log 9}\sqrt{\log n}}\right)} = \frac{\sqrt{3\pi\log n}n^{\log 9 + 1}\frac{1-o(1)}{1+o(1)}}{(1-o(1))28\sqrt{2}\log n(1+o(1))}\\ &= \Omega\left(\frac{n^{\log 9 + 1}}{\sqrt{\log n}}\right).\end{aligned}$$

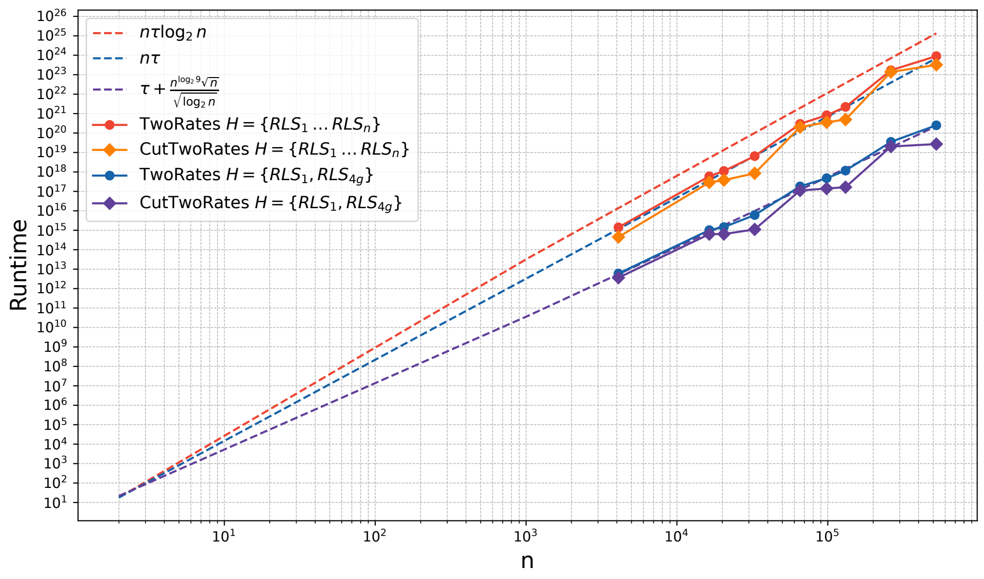

Figure 2: **Experimental results for CUTTWORATES and TWORATES setting** $\tau = n^{3.17}$

Since we have $\Omega\left(\frac{\sqrt{n}}{\log n}\right) 2g$ hurdles, we conclude that the expected runtime is $\Omega\left(\frac{n^{3/2+\log 9}}{\log n\sqrt{\log n}}\right)$. $\square$

6. Experiments

We now present an experimental analysis of the performance of GRG on TWORATES. In all experiments, we plot the average runtime of each GRG setting over 1000 independent runs and increase the problem size n between $2^{12} = 4096$ and $2^{19} - 7 = 524281$. To allow to experiment with large problem sizes we apply the inverse transform sampling method previously used in [18] and sample the expected waiting times to make an improvement of exactly d bits, for all possible values of d and for each operator instead of executing the operator itself.

We plot the results using a logarithmic scale for both the x-axis and y-axis, and use the slope of the experimental line to estimate the asymptotic order of the genuine runtime (i.e. if the runtime is $T = k \cdot n^t$, then $\log T = t\log n + \log k$). All the experiments regarding the CUTTWORATES function are plotted using a diamond shape while the ones regarding the TWORATES function are plotted using dots.

Figure 2 shows the results for $\tau = n^{3.17}$. One may notice that the experimental curve fluctuates a lot. This is because the problem sizes are still too small to reveal the fact, which we use in our theoretical bounds, that any polynomial n^k satisfies $n^k = o(\exp[n^\epsilon])$ where $\epsilon \approx 7.5e10^{-5}$ if we set $\tau = n^{3.17}$.

Figure 3 shows the results for $\tau = n^{3.47}$ for which smaller values of n are required to satisfy the above equation. The experimental curves now become smoother and we see that all the runtimes grow in the same order as our theoretical bounds for realistic problem sizes (i.e. the curves run parallel to the reference lines). Furthermore, the different inclination of the slopes for TWORATES and CUTTWORATES when two operators are used suggest that our bound of $O(n\tau)$ (Theorem 4) for the complete function may not be too pessimistic i.e., the runtime grows faster if the algorithm has to also traverse the final ONEMAX region.

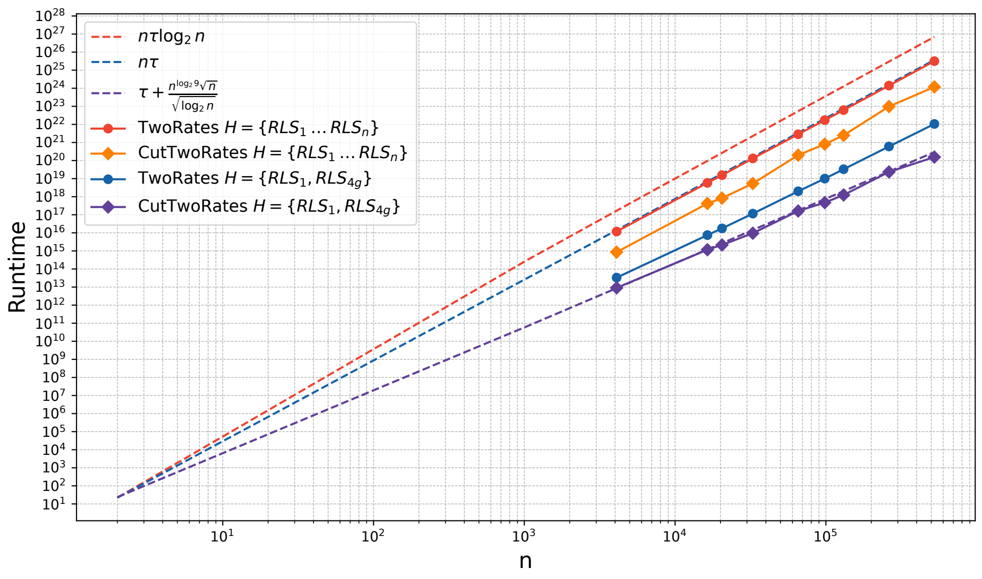

Figure 3: **Experimental results for CutTwoRates and TwoRates setting $\tau = n^{3.47}$**

7. Conclusion and Discussion

We have shown how the GRG SHH can take advantage of its repeated use of its chosen operator to overcome several local optima with basins of attraction of different sizes without having to switch operators once an appropriate one is identified. Compared to the best known upper bound of $\mathcal{O}(n^{4.5})$ for the TwoRates function from the literature, this allows GRG to achieve an expected runtime of $\mathcal{O}(\frac{n^{3.67}}{\sqrt{\log n}})$ to cross the region of local optima equipped with the two optimal operators $H = \{\text{RLS}_1, \text{RLS}_{4g}\}$ and of $\mathcal{O}(n^{4.17} \log n)$ for the whole function when equipped with a complete low-level heuristic set $H = \{\text{RLS}_1, \dots, \text{RLS}_n\}$.

Our result relies on the fact that all the local optima have an even sized Hamming distance from each other. If one length was odd and the other one even i.e., g an odd integer, then it would not be possible to escape from both types of local optima using the same operator. In this case GRG would have to switch operator after overcoming each hurdle, or jump two hurdles at a time, both options leading to considerable slow-downs. This insight suggests the design of a more powerful GRG SHH for multimodal optimisation that is capable of efficiently escaping efficiently from local optima with a larger variety of basins of attraction. In particular our analysis suggests that GRG should be equipped with the possibility of escaping local optima with both odd and even sized basins of attraction with the same operator. One option is to modify the GRG HH such that it applies both RLS_k and RLS_{k+1} after having chosen RLS_k uniformly at random in the operator selection phase. Thus, it would create two offspring in each generation, one by flipping k bits and another by flipping $k+1$ bits (an odd and an even number) and use the best outcome. With such a modification, GRG would be able to optimise TwoRates also with odd values of g in the same asymptotic expected runtimes provided in this paper at the expense of an extra constant factor of 2 in the upper bounds. The resulting extended GRG HH is presented in Algorithm 2.

It is possible that also the Flex-EA algorithm may be able to optimise TwoRates by only applying RLS_{4g} to cross all the hurdles as in our proof strategies. Since the probabilities of choosing

Algorithm 2 Extended GRG Hyper-heuristic

```
1:  Choose $x \in S$ uniformly at random
2:  while stopping conditions not satisfied do
3:      Choose $h \in H$ according to some distribution
4:      $c_t \leftarrow 0$
5:      while $c_t < \tau$ do
6:          $c_t \leftarrow c_t + 1$; $x' \leftarrow h(x)$; $x^\dagger \leftarrow RLS_1(x')$
7:          if $f(x') > f(x)$ then
8:              $c_t \leftarrow 0$; $x \leftarrow x'$
9:          else if $f(x^\dagger) > f(x)$ then
10:             $c_t \leftarrow 0$; $x \leftarrow x^\dagger$
```

each RLS$_k$ in the Flex-EA are bounded from below with a heavy-tailed distribution $\ell_k = \frac{k^{-\beta}}{2N(\beta)}$, the algorithm will select RLS_{4g} with probability $\Theta((1/\log n)^\beta)$ and, once it makes an improvement, make it an active operator in the archive, thus to be re-used with higher probability. Since TwoRates was originally introduced to show the utility of the archive at storing and re-using multiple useful operators in the Flex-EA, this behaviour may have been overlooked. We leave a rigorous analysis for future work.

References

[1] 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, Journal of the Operational Research Society (2013) 1695–1724.

[2] N. Pillay, R. Qu, Hyper-heuristics: theory and applications, Springer, 2018.

[3] J. H. Drake, A. Kheiri, E. Özcan, E. K. Burke, Recent advances in selection hyper-heuristics, European Journal of Operational Research 285 (2) (2020) 405–428.

[4] 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.

[5] 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, AAAI Press, 2019, pp. 2322–2329.

[6] A. Lissovoi, P. S. Oliveto, J. A. Warwicker, When move acceptance selection hyper-heuristics outperform Metropolis and elitist evolutionary algorithms and when not, Artificial Intelligence 314 (2023) 103804.

[7] D. Corus, P. S. Oliveto, D. Yazdani, When hypermutations and ageing enable artificial immune systems to outperform evolutionary algorithms, Theoretical Computer Science 832 (2020) 166–185.

[8] M. A. H. Fajardo, D. Sudholt, Self-adjusting offspring population sizes outperform fixed parameters on the cliff function, Artificial Intelligence 328 (2024) 104061.

[9] B. Doerr, J. F. Lutzeyer, Hyper-heuristics can profit from global variation operators, arXiv preprint arXiv:2407.14237 (2024).

[10] B. Doerr, A. Dremaux, J. Lutzeyer, A. Stumpf, How the move acceptance hyper-heuristic copes with local optima: drastic differences between jumps and cliffs, in: Proceedings of the Genetic and Evolutionary Computation Conference, 2023, pp. 990–999.

[11] 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.

[12] D. Corus, P. S. Oliveto, D. Yazdani, Fast immune system-inspired hypermutation operators for combinatorial optimization, IEEE Transactions on Evolutionary Computation 25 (5) (2021) 956–970.

[13] A. Rajabi, C. Witt, Self-adjusting evolutionary algorithms for multimodal optimization, Algorithmica 84 (2022) 1694–1723.

[14] A. Rajabi, C. Witt, Stagnation detection with randomized local search, Evolutionary Computation 31 (1) (2023) 1–29.

[15] D.-C. 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 Transactions on Evolutionary Computation 22 (3) (2018) 484–497.

[16] D. Antipov, B. Doerr, Runtime analysis of a heavy-tailed $(1 + (\lambda, \lambda)$ genetic algorithm on jump functions, in: Proceedings of the International Conference on Parallel Problem Solving from Nature, PPSN '20, Springer, 2020, pp. 545–559.

[17] 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: Proc. of GECCO 2016, 2016, pp. 645–652.

[18] A. Lissovoi, P. S. Oliveto, J. A. Warwicker, Simple Hyper-Heuristics Control the Neighbourhood Size of Randomised Local Search Optimally for LeadingOnes, Evolutionary Computation 28 (3) (2020) 437–461.

[19] C. Doerr, M. Wagner, Simple on-the-fly parameter selection mechanisms for two classical discrete black-box optimization benchmark problems, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '18, ACM, 2018, pp. 943–950.

[20] 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.

[21] 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, AAAI Press, 2020, pp. 2376–2383.

[22] M. Krejca, C. Witt, A flexible evolutionary algorithm with dynamic mutation rate archive, in: Proc. of GECCO 2024, 2024, pp. 1578–1586.

[23] Y. Ma, P. S. Oliveto, J. A. Warwicker, Random gradient hyper-heuristics can learn to escape local optima in multimodal optimisation, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '25, ACM, 2025.

[24] P. Cowling, G. Kendall, E. Soubeiga, A hyperheuristic approach to scheduling a sales summit, in: Proceedings of the International Conference on Practice and Theory of Automated Timetabling, PATAT '01, Springer, 2001, pp. 176–190.

[25] 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.

[26] V. Chvátal, The tail of the hypergeometric distribution, Discrete Mathematics 25 (3) (1979) 285–287.

[27] B. Doerr, C. Doerr, J. Yang, Optimal parameter choices via precise black-box analysis, in: Proceedings of the Genetic and Evolutionary Computation Conference 2016, GECCO '16, ACM, 2016, pp. 1123–1130.

[28] P. K. Lehre, P. S. Oliveto, Theoretical analysis of stochastic search algorithms, in: R. Martí, P. M. Pardalos, M. G. C. Resende (Eds.), Handbook of Heuristics, Springer International Publishing, Cham, 2018, pp. 849–884.