How the Duration of the Learning Period Affects the Performance of Random Gradient Selection Hyper-Heuristics

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

John Alasdair Warwicker
Institute of Operations Research
Karlsruhe Institute of Technology
76185 Karlsruhe, Germany
john.warwicker@kit.edu

Abstract

Recent analyses have shown that a random gradient hyper-heuristic (HH) using randomised local search (RLS_k) low-level heuristics with different neighbourhood sizes k can optimise the unimodal benchmark function LEADINGONES in the best expected time achievable with the available heuristics, if sufficiently long learning periods τ are employed. In this paper, we examine the impact of the learning period on the performance of the hyper-heuristic for standard unimodal benchmark functions with different characteristics: RIDGE, where the HH has to learn that RLS_1 is always the best low-level heuristic, and ONEMAX, where different low-level heuristics are preferable in different areas of the search space. We rigorously prove that super-linear learning periods τ are required for the HH to achieve optimal expected runtime for RIDGE. Conversely, a sub-logarithmic learning period is the best static choice for ONEMAX, while using super-linear values for τ increases the expected runtime above the asymptotic unary unbiased black box complexity of the problem. We prove that a random gradient HH which automatically adapts the learning period throughout the run has optimal asymptotic expected runtime for both ONEMAX and RIDGE. Additionally, we show experimentally that it outperforms any static learning period for realistic problem sizes.

1 Introduction

One of the main challenges in solving complex and poorly understood optimisation problems is the choice of which heuristic to apply and which related parameters to use. As a result, time consuming trial and error phases are often used to tune the inherent parameters.

Hyper-heuristics have been designed to mitigate such problems by automatically configuring the heuristic and parameters to be employed for the optimisation problem at hand. Selection hyper-heuristics automate the optimisation process by selecting from a set of existing low-level heuristics which one to apply at each decision point during the run. Several examples of their efficiency for solving optimisation problems are available (Cowling, Kendall, and Soubeiga 2001; Burke et al. 2013; 2010; Özcan, Bilgin, and Korkmaz 2008). Theoretical analyses also exist that have confirmed their effectiveness for the optimisation of both unimodal and multimodal benchmark functions (Alanazi and Lehre 2016; Lehre and Özcan 2013; Alanazi and Lehre 2014; Qian, Tang, and Zhou 2016; Lissovoi, Oliveto, and Warwicker 2019b; 2019a; Doerr et al. 2018; Doerr, Doerr, and Yang 2016a). For an overview of theoretical results regarding the performance of hyper-heuristics and other parameter control mechanisms, we refer to the recent survey by Doerr and Doerr (2019).

It is well understood that random gradient hyper-heuristics need to apply low-level heuristics for some time, i.e., the *learning period* τ, to estimate how good or bad the chosen heuristic is for the current region of the search space (Lissovoi, Oliveto, and Warwicker 2019b). In particular, Lissovoi et al. analysed the *Generalised Random Gradient* (GRG) hyper-heuristic that applies a randomly chosen low-level heuristic for τ iterations and restarts the learning period whenever the fitness value is improved. They proved that for the LEADINGONES benchmark function, any constant k, and an appropriately set τ, GRG matches the best-possible expected runtime achievable by using k Randomized Local Search mutation operators with different neighbourhood sizes ($\{\text{RLS}_1, \ldots, \text{RLS}_k\}$) up to lower order terms. In that setting, GRG was shown to have an expected runtime of $((1+\ln(2))/4)n^2 + o(n^2) \approx 0.423n^2 + o(n^2)$ when selecting between two operators, and an expected runtime of $\approx 0.388n^2$ when selecting between at least 18 operators. Such a performance is called *optimal* for the hyper-heuristic, because it is the best that can be achieved with the available low-level heuristics[1]. A consequence of the result is that the hyper-heuristic is also faster than any of its low-level heuristics applied alone for the problem (including the best one, RLS_1, which achieves an expected runtime of $0.5n^2$) . This result required the hyper-heuristic to have a learning period of length $\tau = \omega(n)$ and $\tau \leq (1/k-\epsilon)\cdot n \ln n$ for some constant $\epsilon > 0$.

However, for problems that are not so well-understood, the identification of optimal static values for the learning period may be more prohibitive. Furthermore, the best value for τ may change in different areas of the search space and thus no static value may be optimal throughout the run. To tackle this issue, an *Adaptive Random Gradient* (ARG) hyper-heuristic was recently introduced that incor-

[1]This does not mean that faster runtimes cannot be achieved with different low-level heuristics.

porates a self-adjusting mechanism to adapt the value of τ on the fly (Doerr et al. 2018). It was proven that ARG, when equipped with low-level heuristics that flip 1 or 2 bits with replacement (1BITFLIP and 2BITFLIP), also achieves the optimal expected runtime for LEADINGONES with these heuristics, i.e. $((1+\ln 2)/4)n^2+o(n^2) \approx 0.423n^2+o(n^2)$. Furthermore, it was shown that ARG tracks the best learning period during the run, allowing it to select the *optimal* operator (i.e., the operator with the highest probability of providing a fitness improvement) with high probability in each iteration. ARG also has a parameter, σ, which controls how the learning period τ is adjusted, both by influencing the magnitude of the adjustments to τ, and by requiring σ improvements to be constructed by an operator within a τ-iteration period to re-use the operator for another τ iterations. Theoretical results suggest that σ is considerably more robust than the learning period τ for LEADINGONES (Doerr et al. 2018).

In this paper, we perform an analysis of the impact and use of the learning period on the performance of the GRG hyper-heuristics. We consider two well-studied unimodal benchmark functions from the literature with considerably different characteristics. Our first aim is to understand whether *optimal performance* may be achieved also for other problems, or whether such a result is singular to LEADINGONES. Secondly, we wish to evaluate the robustness of the learning period with respect to different function characteristics (i.e., whether considerably different learning periods may be required for optimal performance on different functions). To address these two questions we consider the hyper-heuristic equipped with a low-level heuristic set of size two and analyse whether the best heuristic is used with high probability in different regions of the search space. We first consider the RIDGE function, where the improvement probabilities of the operators do not change as the search progresses. Hence, one low-level heuristic (i.e., RLS_1) always performs far better than the others. For the hyper-heuristic to excel, a static learning period τ suffices as long as the best low-level heuristic succeeds in τ iterations while the others fail with a sufficiently high probability. Afterwards, we consider the ONEMAX benchmark function, which is much harder for the hyper-heuristic because the success probabilities of each operator vary drastically throughout the run. Flipping many bits has a higher success probability at the beginning of the run, while flipping fewer bits becomes more advantageous as the search progresses. As a result, identifying the best static value[2] for the learning period is difficult as its best possible setting changes during the run.

We first analyse the impact of static values for the parameter τ. Our first conclusion is that for different functions, the best static value for the learning period may be drastically different, even if we restrict the domain to the class of unimodal functions. In particular, GRG equipped with RLS_1 and any other RLS_k operator, with $k = o(n)$, is able to optimise RIDGE in optimal expected time, which is achieved by setting $\tau \geq 5/2 \cdot n \log n, \tau = O\left(n^2/\log n\right)$. On the other hand, setting $\tau \geq n$ is detrimental for ONEMAX, in that the expected runtime of GRG with the low-level heuristic sets $\{\mathrm{RLS}_1, \mathrm{RLS}_2\}$ and $\{\mathrm{RLS}_1, \mathrm{RLS}_3\}$ is asymptotically worse than that of the unary unbiased black box complexity of ONEMAX (Lehre and Witt 2012). Hence, it is asymptotically slower than RLS_1 used on its own (i.e., $n \ln n - 0.1159...n \pm o(n)$, Witt 2014). The best setting for τ is in fact shown to be $\tau = o(\log n)$, giving a performance for both sets of heuristics that is at most a constant factor worse than RLS_1, yet still asymptotically optimal.

While the latter result only provides an upper bound on the expected runtime of GRG, the analysis suggests that a dynamic learning period may lead to better performance for ONEMAX (i.e., to achieve asymptotically optimal runtime, τ should be large enough such that RLS_1 succeeds with high probability, yet other heuristics fail). In any case, since considerably different static values for τ are required for the best achievable performance on the two problems, we consider ARG which adapts the learning period on the fly. We rigorously prove that ARG optimises both RIDGE and ONEMAX in optimal asymptotic expected runtime.

We complement the theoretical results with experiments for various problem sizes for ONEMAX and RIDGE. For GRG, the experiments confirm that as τ increases, the runtime also increases. However, setting τ to a small value is preferable to having no learning period at all, confirming the benefits of the learning period. For sufficiently large problem sizes, ARG is shown to be able to beat GRG experimentally for all tested static values of τ. The experiments also imply that σ is much more robust than τ and thus confirm that ARG is preferable in practice.

The rest of this paper is structured as follows. In the next section, we introduce the benchmark functions, the hyper-heuristics and the mathematical tools we use in our analysis. The performance analysis of GRG with respect to τ is presented in Section 3 and the performance analysis of ARG is presented in Section 4. Complementary experiments are presented in Section 5. We conclude the paper with a discussion of the presented results and avenues for future work.

Due to space constraints, this extended abstract omits many formal proofs.

2 Preliminaries

In this section, we introduce the unimodal benchmark functions, the selection hyper-heuristics, and the mathematical techniques considered in the rest of the paper.

2.1 Unimodal Benchmark Functions

The first benchmark function we consider is the standard RIDGE function (Quick, Rayward-Smith, and Smith 1998) which consists of a gradient of increasing fitness as the number of 0-bits in the bit-string increases, and a gradient of increasing fitness as the number of prefix 1-bits increases as long as the solution is of the form 1^i0^{n-i}, leading to a global optimum at 1^n:

$$\textsc{Ridge}(x) := \begin{cases} n+i & \text{if } x = 1^i0^{n-i} \text{ for } i \in \mathbb{N}_0 \\ n - \sum_{i=1}^{n} x_i & \text{otherwise.} \end{cases}$$

[2] That is, a fixed value of τ that, when used throughout the run, minimises the expected runtime of the hyper-heuristic.

Algorithm 1 Generalised Random Gradient Hyper-heuristic

Choose $x \in S$ uniformly at random
while stopping conditions not satisfied **do**
  Choose $h \in H$ uniformly at random
  $c_t \leftarrow 0$
  while $c_t < \tau$ **do**
    $c_t \leftarrow c_t + 1$; $x' \leftarrow h(x)$
    if $f(x') > f(x)$ **then**
      $c_t \leftarrow 0$; $x \leftarrow x'$

To isolate hyper-heuristic performance on the ridge region of the search space, we consider algorithms initialised at 0^n. As a consequence, no offspring with fitness smaller than n are ever accepted, and the improvement probability of each low-level heuristic remains the same throughout almost all of the search space.

In order for a local search operator RLS_k to increase the fitness, it needs to flip exactly the first k consecutive 0-bits, an event that occurs with probability $\Theta(n^{-k})$ when at least $k = O(1)$ 0-bits are present. Hence, RLS_1 is the mutation operator with the highest success probability (i.e., $1/n$) and applying this operator throughout the run yields the best expected runtime, namely $E[T_{\text{RLS}_1}] = \sum_{i=0}^{n-1} n = n^2$.

The other benchmark function we consider is ONEMAX (OM for brevity). The goal is to minimise the Hamming distance to a hidden bit-string (i.e., the optimum). For analytical convenience, the optimum is usually set at 1^n, such that $\text{OM}(x) = \sum_{i=1}^{n} x_i$. The function exhibits the common hillclimbing property expected to appear in optimisation problems, i.e., that improving solutions are harder to find as the global optimum is approached. This class of functions is known to be the easiest among all functions with a unique global optimum for unary unbiased black box algorithms (Lehre and Witt 2012). Amongst these, the best is RLS_1, which has an expected runtime of $E[T_{\text{RLS}_1}] = n \ln n - 0.1159...n \pm o(n)$ (Witt 2014).

Since we initialise all algorithms at 0^n on RIDGE, we opt to do so also for ONEMAX. However, for ONEMAX, we expect the difference in results to be minor, as any RLS_k operator with a constant $k \geq 1$ would be able to construct an individual with $n/2$ 1-bits in expected $O(n)$ iterations.

2.2 Selection Hyper-heuristics

Throughout the paper, we refer to the expected runtime of a hyper-heuristic that matches the leading constant in the best expected runtime that can be achieved with the given low-level heuristic set H as the *optimal* expected runtime.

The Generalised Random Gradient (GRG) hyper-heuristic was analysed (Lissovoi, Oliveto, and Warwicker 2019b) as an extension of the classical Random Gradient hyper-heuristic (Cowling, Kendall, and Soubeiga 2001). It applies a randomly chosen low-level heuristic for a learning period of τ iterations. As soon as the heuristic finds an improving solution, a new period of length τ is started. If by the end of a learning period an improvement has not been found, then GRG chooses a new low-level heuristic at random. Algorithm 1 shows the pseudocode for GRG.

Algorithm 2 Adaptive Random Gradient Hyper-Heuristic

$\tau \leftarrow \tau_0$
Choose $x \in S$ uniformly at random
while stopping conditions not satisfied **do**
  Choose $h \in H$ uniformly at random
  $c_t \leftarrow 0$; $c_s \leftarrow 0$
  while $c_t < \tau$ **do**
    $c_t \leftarrow c_t + 1$; $x' \leftarrow h(x)$
    if $f(x') > f(x)$ **then**
      $c_s \leftarrow c_s + 1$; $x \leftarrow x'$
    if $c_s \geq \sigma$ **then**
      $c_s \leftarrow 0$; $c_t \leftarrow 0$
      $\tau \leftarrow \tau \cdot F^{-1/\sigma^2}$
  $\tau \leftarrow \tau \cdot F^{1/\sigma}$

Since the optimal value of the learning period may change throughout the search space, GRG has recently been equipped with a mechanism to automatically adapt the duration of the learning period during the run (Doerr et al. 2018). As a result, the value of τ does not need to be manually set in advance. The resulting hyper-heuristic was named Adaptive Random Gradient (ARG), and its pseudocode is shown as Algorithm 2. The adaptive mechanism uses a $1 - 1/\sigma$-rule, which increases the learning period ($\tau \leftarrow \tau \cdot F^{1/\sigma}$) when the low-level heuristics construct fewer than σ fitness improvements during a learning period, while it decreases it by a smaller amount ($\tau \leftarrow \tau \cdot F^{-1/\sigma^2}$) if they succeed at improving solution fitness at least σ times.

It has previously been proved that ARG with $H = \{1\text{BITFLIP}, 2\text{BITFLIP}\}$[3] achieves optimal expected runtime for LEADINGONES (i.e., $(1 + o(1)) \cdot (1 + \ln 2)/4 \cdot n^2 \approx 0.423n^2 + o(n^2)$), with $\sigma = \Omega(\log^4 n) \cap o(\sqrt{n/\log n})$ (Doerr et al. 2018).

2.3 Mathematical Analysis Tools

The following drift analysis theorems are often used to bound the expected runtime of a randomised search heuristic by considering its drift (i.e, the expected decrease in distance from the optimum in each step). We use the Hamming distance as a measure of distance to the optimum and write $\Delta(i) = E(X^{(t)} - X^{(t+1)} \mid X^{(t)} = n - i)$ to refer to the drift conditioned on the parent solution having a Hamming distance of $n - i$ to the optimum.

The following additive drift theorem provides bounds on the expected runtime given 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 $S \in \mathbb{R}_0^+$ and let T be the random variable that denotes the first point in time for which $X^{(t)} = 0$. If there exist $\delta_u \geq \delta_l > 0$ such that for all $t \geq 0$ we have*

$$\delta_u \geq E(X^{(t)} - X^{(t+1)} \mid X^{(t)}) \geq \delta_l,$$

[3] Unlike RLS_2, 2BITFLIP heuristic flips two bits chosen uniformly at random *with* replacement.

then the expected optimisation time satisfies

$$E(X^{(0)})/\delta_u \leq E(T) \leq E(X^{(0)})/\delta_l.$$

If the drift changes considerably throughout the process, then the variable drift theorem may provide sharper upper bounds on the expected runtime.

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

$$E(T \mid X^{(0)}) \leq \frac{x_{\min}}{h(x_{\min})} + \int_{x_{\min}}^{X^{(0)}} \frac{1}{h(x)}\, dx.$$

3 Static Learning Periods

In this section, we identify the optimal static learning periods for the Generalised Random Gradient (GRG) hyper-heuristic for RIDGE and for ONEMAX, when it is equipped with two low-level heuristics. In Subsection 3.1, we prove that for RIDGE, the optimal value for τ is super-linear, leading to the optimal expected runtime. In Subsection 3.2, we show that for ONEMAX, such a value for the learning period leads to an expected runtime that is asymptotically worse than that of one of its low level heuristics (i.e., RLS_1). On the other hand, setting the learning period to $\tau = o(\log n)$ leads to optimal asymptotic expected runtime.

3.1 RIDGE

In this subsection we consider GRG with the low-level heuristic set $H = \{\text{RLS}_1, \text{RLS}_k\}$ where $k = o(n)$.

We first show that if the learning period of GRG is too short, then neither low-level heuristic is able to succeed reliably and the hyper-heuristic devolves to simply using both heuristics uniformly at random.

Theorem 3. *Starting at 0^n, the expected runtime of the Generalised Random Gradient hyper-heuristic for* RIDGE *with $H = \{\text{RLS}_1, \text{RLS}_k\}$ ($k \geq 2$, $k = o(n)$) and $\tau = o(n)$ is $(1 \pm o(1)) \cdot 2n^2$.*

In Theorem 4, we prove that if the learning period of GRG is set sufficiently high, yet not excessively so, then the hyper-heuristic can learn to use RLS_1 in each iteration with high probability and thus has optimal expected runtime for RIDGE up to lower order terms. Intuitively, the proof exploits that while RLS_k usually fails after being chosen, RLS_1 can usually succeed multiple times before failing, and is thus used much more often.

Theorem 4. *Starting at 0^n, the expected runtime of the Generalised Random Gradient hyper-heuristic for* RIDGE *with $H = \{\text{RLS}_1, \text{RLS}_k\}$ and $\tau \geq (1+\epsilon) n \ln n$, $\tau = O\left(n^2/\log n\right)$ is $(1 + o(1)) \cdot n^2$ for any $k = o(n)$ and any constant $\epsilon \geq 0$.*

3.2 ONEMAX

In this subsection, we consider GRG using $H = \{\text{RLS}_1, \text{RLS}_3\}$, i.e., the two best low-level heuristics for ONEMAX[4]. In Theorem 5, we prove that if τ is too large, then the suboptimal operator is used too often, leading to poor performance.

Theorem 5. *Starting at 0^n, the expected runtime of the Generalised Random Gradient hyper-heuristic for* ONEMAX *with $H = \{\text{RLS}_1, \text{RLS}_3\}$ and $\tau \geq n$ is $\Omega\left(n\sqrt{n}/(\sqrt{\log n})\right) = \omega(n \log n)$.*

Proof. The probability of a fitness improvement (of any amount) of RLS_3 in one iteration for ONEMAX, at the state $\text{OM}(x) = i$, denoted $p_3(i)$, is (Doerr, Doerr, and Yang 2016b, Section 4.1),

$$p_3(i) = \frac{(n-i)(n-i-1)(n-i-2+3\cdot i)}{n(n-1)(n-2)}.$$

This is a monotonically decreasing function with respect to i (as $p_3(0) = p_3(1) = 1$ and $\frac{dp_3(i)}{di} \leq 0$ for $1 \leq i \leq n-1$). As the algorithm progresses towards the global optimum, it is harder for the heuristics (including RLS_3) to find improving solutions and the probability of finding an improvement decreases. Hence, the probability of improvement of RLS_3 at $\text{OM}(x) = n - \sqrt{n \ln n}$ (i.e., $\frac{\sqrt{n\ln n}(\sqrt{n\ln n}-1)(3n-2\sqrt{n\ln n}-2)}{n(n-1)(n-2)} \geq 5/2 \cdot \frac{\ln n}{n}$) gives a lower bound on the probability of improvement of RLS_3 when $\text{OM}(x) \leq n - \sqrt{n \ln n}$. The probability that RLS_3 fails to find an improvement within $\tau \geq n$ iterations while $\text{OM}(x) \leq n - \sqrt{n \ln n}$ is at most

$$(1 - p_3(i))^{\tau} \leq \left(1 - \frac{5\ln n}{2n}\right)^n \leq \left(1 - \frac{5\ln n}{2n}\right)^{\frac{2n}{5\ln n}\cdot\frac{5\ln n}{2}}$$
$$\leq \exp(-(5/2)\ln n) = n^{-5/2}.$$

Let G be the event that the time between improvements constructed by RLS_3 is below τ until a solution with fitness of at most $n - \sqrt{n \ln n}$ is constructed. Equivalently, G is the event that RLS_3, once picked, does not fail before reaching a fitness of at least $n - \sqrt{n \ln n}$. In the worst case, this requires RLS_3 to succeed in at most $n - \sqrt{n \ln n}$ consecutive periods of τ iterations, and thus

$$P(G) \geq (1 - n^{-5/2})^{n - \sqrt{n \ln n}} > 1 - n^{-3/2} = 1 - o(1).$$

If only RLS_3 is used, we can bound the expected time T' to reach a fitness of $\text{OM}(x) = n - \sqrt{n \ln n}$ using the lower bound variant of the variable drift theorem (Doerr, Fouz, and Witt 2011, Theorem 7). Let $X^{(t)} = \max\{0, n - \text{OM}(x) - \sqrt{n \ln n}\}$ and let $c(X^{(t)}) = X^{(t)} - 3$. The drift in one step of RLS_3 for OM is $h(c(X^{(t)})) = \frac{3(n-i)(n-i-1)}{n(n-1)}$ per (Doerr, Doerr, and Yang 2016b). Let $x_{\min} = 1$. Hence, $X^{(0)} =$

[4]This pair of heuristics, when applied optimally, has a smaller expected runtime than any other $\{RLS_a, RLS_b\}$ pair.

$n - \sqrt{n \ln n}$ and $h(x_{\min}) = \Theta(\sqrt{n \log n}/n^2)$ and,

$$\begin{aligned} E(T') &\geq \Omega\left(\frac{n^2}{\sqrt{n \log n}}\right) + \int_1^{n-\sqrt{n \ln n}} \frac{1}{h(X^{(t)})}\, dX^{(t)} \\ &= \Omega\left(\frac{n^2}{\sqrt{n \log n}}\right) + \int_0^{n-\sqrt{n \ln n}-1} \frac{n(n-1)}{3(n-i)(n-i-1)}\, di \\ &= \Omega\left(n\sqrt{n}/\sqrt{\log n}\right). \end{aligned}$$

Additionally, a matching upper bound can be obtained by applying Theorem 2:

$$\begin{aligned} E(T') &\leq \frac{x_{\min}}{h(x_{\min})} + \int_1^{n-\sqrt{n \ln n}} \frac{1}{h(x)} dx \\ &\leq O\left(\frac{n\sqrt{n}}{\sqrt{\log n}}\right) + \int_0^{n-\sqrt{n \ln n}-1} \frac{n(n-1)}{3(n-i)(n-i-1)} di \\ &\leq O\left(\frac{n\sqrt{n}}{\sqrt{\log n}}\right) + \frac{1}{3}\frac{n\sqrt{n}}{\sqrt{\ln n}} = O\left(\frac{n\sqrt{n}}{\sqrt{\log n}}\right). \end{aligned}$$

We apply the law of total expectation to obtain a bound on the expected time for RLS_3 to reach a fitness of $n - \sqrt{n \ln n}$ conditional on the event G occurring (i.e. on RLS_3 not failing before reaching the required fitness):

$$\begin{aligned} E(T') &= E(T' \mid G)P(G) + E(T' \mid \neg G)P(\neg G), \quad \Leftrightarrow \\ E(T' \mid G) &= \Big(E(T') - E(T' \mid \neg G)P(\neg G)\Big)/P(G) \\ &\geq E(T') - E(T' \mid \neg G)P(\neg G) \\ &\geq E(T') - (\tau + E(T'))P(\neg G) = \Omega(E(T')), \end{aligned}$$

as $E(T') = \Omega\left(\frac{n\sqrt{n}}{\sqrt{\log n}}\right)$, $P(\neg G) \leq n^{-3/2}$ already for $\tau = n$, and $E(T' \mid \neg G) \leq \tau + E(T')$ (i.e., adding τ iterations where no progress is made satisfies the gap requirement of $\neg G$, allowing the rest of the optimisation to proceed without any additional constraints on when improvements can occur).

As RLS_3 is selected initially with probability $1/2$, the expected runtime of GRG with $\tau \geq n$ for ONEMAX is at least,

$$E(T) \geq 1/2 \cdot P(G) \cdot E(T' \mid G) = \Omega\left(\frac{n\sqrt{n}}{\sqrt{\log n}}\right),$$

by the law of total expectation. □

An equivalent result for $H = \{\mathrm{RLS}_1, \mathrm{RLS}_2\}$ can be obtained by following the same proof strategy.

We now analyse the expected runtime of GRG when τ is sub-linear. In this setting, both operators fail reliably once the hyper-heuristic reaches the difficult portion of the search space, and the frequent failures produce an almost even split between using the two low-level heuristics.

Theorem 6. *Starting at 0^n, the expected runtime of the Generalised Random Gradient hyper-heuristic for* ONEMAX *with $H = \{\mathrm{RLS}_1, \mathrm{RLS}_3\}$ and $\tau = o(n), \tau > 0$ is at most $(1 + o(1)) \cdot (6\tau n + 2n \ln n)$.*

An equivalent result for $H = \{\mathrm{RLS}_1, \mathrm{RLS}_2\}$ can be obtained by following the same proof strategy.

Furthermore, if an operator is chosen at random in *every* iteration (akin to setting $\tau = 0$), the expected runtime is $(1+o(1)) \cdot 2n \ln n$ by an application of the variable drift theorem. Using short learning periods, i.e. $\tau = o(\log n)$, minimises the leading term in the upper bound of Theorem 6. In Section 5, empirical results show that short learning periods of more than 1 iteration are beneficial in practice.

4 Adaptive Learning Periods

In the previous section, we proved that considerably different static values for τ are required for the best achievable performance on the two considered benchmark functions. In this section, we study whether the Adaptive Random Gradient (ARG) hyper-heuristic achieves optimal asymptotic expected performance for both problems. For our purposes, we consider ARG equipped with the two individually best low-level heuristics for each problem. In this setting, we have provided contrasting results for both problems for each range of the learning period τ. We prove that optimal performance up to the leading constant in the expected runtime is achieved for RIDGE, and asymptotically optimal performance is achieved for ONEMAX.

4.1 RIDGE

As before, we begin the analysis of the impact of adaptive learning periods for RIDGE. The behaviour of ARG on RIDGE is substantially similar to its behaviour on LEADINGONES, which has been analysed previously (Doerr et al. 2018). Throughout the entire optimisation process, the optimal and non-optimal operators have sufficiently different probabilities of producing an improvement to be distinguished by ARG's learning mechanism, and the optimal operator has the highest probability of producing an improvement. For RIDGE, the analysis of Doerr et al. can be simplified somewhat as the probability of RLS_k producing an improvement does not change until fewer than k 0-bits remain in the solution, and so the optimal operator does not change throughout the run. The absence of free-rider bits, causing improvements by RLS_1 to always increase fitness by 1 and improvements by RLS_k to always increase fitness by k, does not compensate sufficiently for the much lower improvement probability of RLS_k. By making minor adaptations to the analysis of ARG for LEADINGONES, we can achieve an equivalent result for RIDGE. We remark that using $H = \{\mathrm{RLS}_1, \mathrm{RLS}_2\}$ allows us to directly reuse some analysis of this heuristic set on the LEADINGONES problem (Doerr et al. 2018), but we do expect that substantially similar bounds can be made for $H = \{\mathrm{RLS}_1, \mathrm{RLS}_k\}$ for any $k = O(1)$, as well as for larger sets H.

Theorem 7. *The expected runtime of the Adaptive Random Gradient hyper-heuristic for* RIDGE *with $H = \{\mathrm{RLS}_1, \mathrm{RLS}_2\}$, $F = 1.5$, $\tau_0 = 1$, and $\sigma = \Omega(\log^4 n) \cap o(\sqrt{n/\log n})$, when initialised at $x_0 = 0^n$ is $(1+o(1)) \cdot n^2$.*

4.2 ONEMAX

Theorem 8 shows that ARG optimises ONEMAX in the optimal asymptotic expected runtime achievable by unary unbiased black box heuristics (including its low level heuristics).

Theorem 8. *Starting at 0^n, the expected runtime of the Adaptive Random Gradient hyper-heuristic for* ONEMAX

with $H = \{\text{RLS}_1, \text{RLS}_3\}$, *for* $\tau_0 = 1$, $\sigma \in \Omega\left((\log n)^{3/4}\right) \cap o(\log n)$ *and* $F = 1 + \frac{1}{\sqrt{\ln n}}$, *is* $E[T_{\text{ARG}}] = O(n \log n)$.

The statement of Theorem 8 requires $\sigma \in \Omega((\log n)^{3/4}) \cap o(\log n)$. While this is a narrow range of values to select from, this is necessary only for the proof to hold. The experimental results presented in Section 5 suggest that the parameter σ is a lot more robust in practice, since a wide range of σ values lead to a performance for ONEMAX that is better than most naïve choices of static τ (e.g., $\tau = 1$ or $\tau = n$), while the best σ outperforms the best τ for sufficiently large problem sizes (i.e., $n \geq 10^5$). For technical reasons, our result requires that F decreases toward 1 as the problem size n increases; experimental results presented in Section 5 show that the standard choice of $F = 1.5$, as suggested by (Doerr, Doerr, and Ebel 2015) and used in Theorem 7 for RIDGE, works even better in practice.

We now present an outline of the proof of Theorem 8, which follows a similar structure to the proof of Theorem 3.1 in (Doerr et al. 2018).

We call a period of at most τ iterations in which a mutation operator produces σ ONEMAX improvements by mutation a *successful phase* and call a period of τ iterations in which a mutation operator produces less than σ ONEMAX improvements a *failed phase*. We bound $T_{\text{ARG}} = T_{\text{R}} + T_{\text{S}} + T_{\text{NS}} + T_{\text{F}}$ by bounding each of the four components:

- T_{R}, the number of iterations spent taken until the candidate solution has a fitness of at least $\text{OM}(x) = 3n/4$ for the first time.
- T_{S}, the number of iterations spent in successful phases applying RLS_1 when $\text{OM}(x) \geq 3n/4$,
- T_{NS}, the number of iterations spent in successful phases applying RLS_3 when $\text{OM}(x) \geq 3n/4$.
- T_{F}, the number of iterations spent in failed phases when $\text{OM}(x) \geq 3n/4$.

To prove the theorem, we will bound $E[T_{\text{S}}] \leq E[T_{\text{RLS}_1}] = n \ln(n) \pm O(n)$ and show that the expected values of the other contributing terms are at most $O(n \log n)$.

Firstly, we prove that $E[T_R] = O(n)$ in Lemma 9.

Lemma 9. *The expected time for the Adaptive Random Gradient hyper-heuristic with* $H = \{\text{RLS}_1, \text{RLS}_3\}$ *to reach a fitness value of* $\text{ONEMAX}(x) \geq 3n/4$ *is* $O(n)$.

We then define a threshold $\tau_{\max}(i)$, which the learning period τ with high probability does not exceed while the solution contains i 1-bits:

$$\tau_{\max}(i) := \frac{3}{2} \cdot \sigma \cdot \frac{1}{p_{\text{opt}}(i)} = \frac{3}{2} \cdot \sigma \cdot \frac{n}{n-i},$$

where $p_{\text{opt}}(i)$ is the improvement probability of the RLS_1 operator at the state $\text{OM}(x) = i$. To bound $E[T_{\text{NS}}]$ and $E[T_{\text{F}}]$, we will prove the following:

1. While $\text{OM}(x) \geq 3n/4$, and $\tau < \tau_{\max}(i)$, RLS_3 fails a phase with probability $1 - \exp(-\Omega(\sigma))$ (Lemma 10).
2. With high probability, τ remains below $\tau_{\max}(i)$ throughout the optimisation process (Lemma 11).

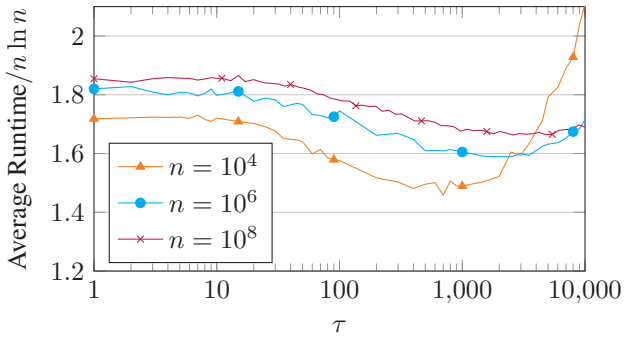

Figure 1: The runtime of Generalised Random Gradient for ONEMAX, with $H = \{\text{RLS}_1, \text{RLS}_3\}$, as τ increases through constant values, for three different problem sizes.

Lemma 10. *Consider a run of Adaptive Random Gradient for* ONEMAX, *with* $H = \{\text{RLS}_1, \text{RLS}_3\}$. *While* $\text{ONEMAX}(x) = i \geq 3n/4$ *and* $\tau < \tau_{\max}(i)$, RLS_3 *produces at least* σ *improvements within* τ *iterations with probability at most* $\exp(-\Theta(\sigma))$.

Lemma 11. *With probability at least* $1 - n^{-c'}$ *for any* $c' > 0$, *Adaptive Random Gradient with* $H = \{\text{RLS}_1, \text{RLS}_3\}$ *finds the global optimum before* $\tau \geq \tau_{\max}(i)$ *occurs, where* i *is the* ONEMAX *value of the ancestor individual in any given iteration.*

We can now bound $E[T_{\text{NS}}]$ (Lemma 12) and $E[T_{\text{F}}]$ (Lemma 13), conditioned on the event that $\tau < \tau_{\max}(i)$ holds throughout the optimisation process (i.e., L_4 holds).

Lemma 12. *While* $\tau < \tau_{\max}(i)$ *and* $\text{ONEMAX}(x) \geq 3n/4$, *the expected number of iterations Adaptive Random Gradient with* $H = \{\text{RLS}_1, \text{RLS}_3\}$ *spends in successful phases when applying* RLS_3 *is* $o(n \log n)$.

Lemma 13. *While* $\tau < \tau_{\max}(i)$ *and* $\text{ONEMAX}(x) \geq 3n/4$, *the expected number of iterations Adaptive Random Gradient with* $H = \{\text{RLS}_1, \text{RLS}_3\}$ *spends in failed phases is* $O(n \log n)$.

Finally, we present an unconditional bound on the expected runtime of ARG for ONEMAX (i.e., $\tau \geq \tau_{\max}(i)$ might occur at some point throughout the optimisation process). The proof of the following lemma largely matches a similar result for LEADINGONES (Doerr et al. 2018, Lemma 3.3), while the prerequisite lemmata also hold.

Lemma 14. *Consider a run of Adaptive Random Gradient with* $H = \{\text{RLS}_1, \text{RLS}_3\}$, *started with an arbitrary initial search point* x *and an arbitrary initial period length* $\tau_0 \leq n^3$, *for* ONEMAX. *Let* T *be the number of fitness evaluations performed up to the point when for the first time the optimal solution is evaluated. Then* $E[T] = O(\sigma n^3)$.

Theorem 8 is proved by combining these results.

5 Experimental Analysis

We complement our theoretical results with an experimental analysis which fills in some gaps left open by the theorems.

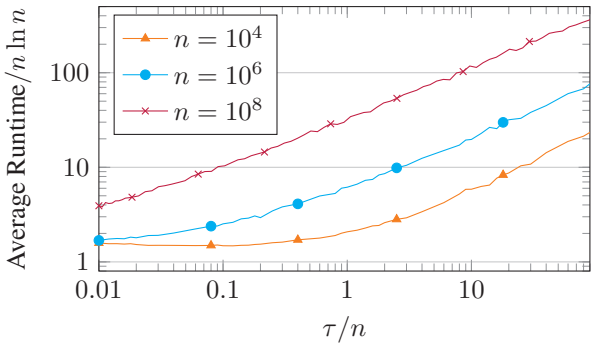

Figure 2: The runtime of Generalised Random Gradient for ONEMAX, with $H = \{\text{RLS}_1, \text{RLS}_3\}$, as τ increases through linear values, for three different problem sizes.

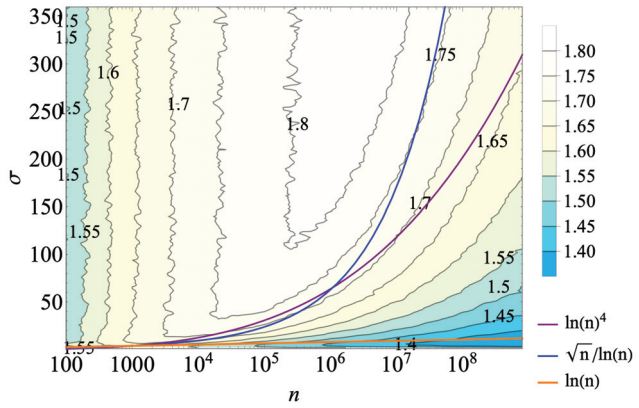

Figure 3: The runtime of Adaptive Random Gradient with $H = \{\text{RLS}_1, \text{RLS}_3\}$ for ONEMAX, for various problem sizes n and parameter values σ, with contour values showing the average observed runtime over 1000 runs divided by $n \ln n$. Several strategies for setting σ depending on n are shown as line plots.

Figures 1 and 2 show the runtime of GRG (initialised at 0^n) for ONEMAX with various values of τ fixed throughout the run, averaged over 500 runs for each τ value. The most significant term in the theoretical upper bound on the runtime presented in Theorem 6 is minimised by choosing $\tau = o(\log n)$, and the experiments confirm that short, but increasing with respect to n, static learning periods produce the best results: using e.g. $\tau = 2500$ results in an average runtime of $1.589n \ln n$ for $n = 10^6$, while $\tau = 1$ resulted in an average runtime of $1.820n \ln n$. While the average runtimes approach our theoretical upper bound, it is unclear whether the leading constant in this bound is tight.

ARG is able to adapt the learning period τ during the run, with the adaptation speed determined by the parameter σ. Our theorem for RIDGE holds for larger σ intervals than those required for our proof for ONEMAX to hold. While the following sets of experiments confirm that the best parameter values for each problem are those given by our theorems already for small problem sizes, they also show that

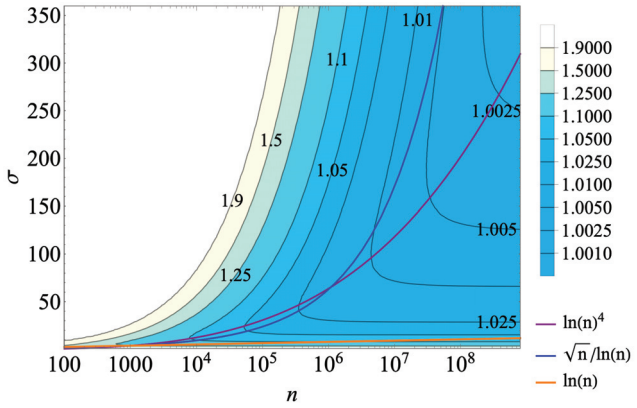

Figure 4: The runtime of Adaptive Random Gradient with $H = \{\text{RLS}_1, \text{RLS}_2\}$ for RIDGE, for various problem sizes n and parameter values σ, with contour values showing the average observed runtime over 1000 runs divided by n^2. Several strategies for setting σ depending on n are shown as line plots.

ARG performs extremely well for both problems for all the considered values for σ.

Figure 3 shows the average runtime of ARG (initialised at 0^n) for ONEMAX for various problem sizes and choices of σ (using $F = 1.5$ for which our theory for RIDGE holds, which performs slightly better for ONEMAX than the value of $F = 1 + 1/\sqrt{\ln n}$ used by Theorem 8), as well as curves showing the boundaries of the policies suggested by our theoretical results for setting σ (with leading constants chosen to achieve $\sigma = 4$ at $n = 10^3$).

The parameter σ is quite robust: the observed average runtime of ARG for ONEMAX remains below $1.86n \ln n$ for all considered values of n and σ. The optimal choice of σ outperforms the best static choice of τ, while for large enough n, any tested value of σ outperforms the best static choice of τ: thus, even if σ is chosen based on what is optimal on RIDGE, ARG still outperforms the optimal static choice of τ for ONEMAX.

Figure 4 shows the performance of ARG on the RIDGE problem when using the two best-performing low-level heuristics. ARG is able to adapt the learning period τ adequately during the run, achieving leading constants in the runtime that are extremely close to the optimal (i.e., 1) for all tested values of σ. In particular, ARG also achieves close-to-optimal performance for RIDGE when σ is set to values which have the best performance for ONEMAX.

6 Conclusion

In this paper, we analysed the impact of the learning period τ on the performance of a random gradient hyper-heuristic equipped with two low-level heuristics. Two unimodal functions with different characteristics, i.e., RIDGE and ONEMAX, were considered for the analysis.

We first considered static values of τ. We rigorously proved that while super-linear values of τ lead to expected optimal performance of the Generalised Random Gradient

hyper-heuristic for RIDGE (up to lower order terms), the same parameter setting for ONEMAX leads to suboptimal asymptotic expected runtimes (i.e., $\omega(n \log n)$). We then proved that a small value for τ is preferable for ONEMAX, providing $O(n \log n)$ expected runtime as desired.

Due to this discrepancy, we analysed whether the Adaptive Random Gradient hyper-heuristic can adapt τ during the run effectively for both problems. We proved that it achieves optimal expected asymptotic runtime for ONEMAX and proved that it can optimise RIDGE in optimal expected runtime up to the leading constant. An experimental analysis confirms the robustness of the adaptation speed σ compared to that of the learning period τ for realistic problem sizes. It remains an open problem whether ARG achieves optimal runtime on ONEMAX, or whether a more complex learning mechanism such as reinforcement learning, which has been proven to do so (Doerr, Doerr, and Yang 2016b), is necessary.

Future work should evaluate the impact of the learning period on other problems, including ones from classical combinatorial optimisation.

Acknowledgements This work was supported by the EPSRC under Grant No. EP/M004252/1.

References

Alanazi, F., and Lehre, P. K. 2014. Runtime analysis of selection hyper-heuristics with classical learning mechanisms. In *Proceedings of IEEE Congress on Evolutionary Computation*, CEC '14, 2515–2523. IEEE.

Alanazi, F., and Lehre, P. K. 2016. Limits to learning in reinforcement learning hyper-heuristics. In *Evolutionary Computation in Combinatorial Optimization*, EvoCOP '16, 170–185. Springer.

Burke, E. K.; Hyde, M.; Kendall, G.; Ochoa, G.; Özcan, E.; and Woodward, J. R. 2010. *A Classification of Hyper-heuristic Approaches*. Springer. 449–468.

Burke, E.; Gendreau, M.; Hyde, M.; Kendall, G.; Ochoa, G.; Özcan, E.; and Qu, R. 2013. Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society* 64(12):1695–1724.

Cowling, P.; Kendall, G.; and Soubeiga, E. 2001. A hyper-heuristic approach to scheduling a sales summit. In *Practice and Theory of Automated Timetabling*, PATAT '01, 176–190. Springer.

Doerr, B., and Doerr, C. 2019. Theory of parameter control for discrete black-box optimization: Provable performance gains through dynamic parameter choices. In Doerr, B., and Neumann, F., eds., *Theory of Evolutionary Computation: Recent Developments in Discrete Optimization (to appear)*. Springer. CoRR, abs/1804.05650.

Doerr, B.; Lissovoi, A.; Oliveto, P. S.; and Warwicker, J. A. 2018. On the runtime analysis of selection hyper-heuristics with adaptive learning periods. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '18. ACM.

Doerr, B.; Doerr, C.; and Ebel, F. 2015. From black-box complexity to designing new genetic algorithms. *Theoretical Computer Science* 567:87 – 104.

Doerr, B.; Doerr, C.; and Yang, J. 2016a. k-bit mutation with self-adjusting k outperforms standard bit mutation. In *Proceedings of the International Conference on Parallel Problem Solving from Nature*, PPSN '16, 824–834. Springer.

Doerr, B.; Doerr, C.; and Yang, J. 2016b. Optimal parameter choices via precise black-box analysis. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '16, 1123–1130. ACM.

Doerr, B.; Fouz, M.; and Witt, C. 2011. Sharp bounds by probability-generating functions and variable drift. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '11, 2083–2090. ACM.

He, J., and Yao, X. 2001. Drift analysis and average time complexity of evolutionary algorithms. *Artificial Intelligence* 127(1):57–85.

Johannsen, D. 2010. *Random combinatorial structures and randomized search heuristics*. Ph.D. Dissertation, Universität des Saarlandes, Saarbrücken.

Lehre, P. K., and Özcan, E. 2013. A runtime analysis of simple hyper-heuristics: To mix or not to mix operators. In *Proceedings of Foundations of Genetic Algorithms XII*, FOGA '13, 97–104. ACM.

Lehre, P. K., and Witt, C. 2012. Black-box search by unbiased variation. *Algorithmica* 623–642.

Lissovoi, A.; Oliveto, P. S.; and Warwicker, J. A. 2019a. On the time complexity of algorithm selection hyper-heuristics for multimodal optimisation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, AAAI '19, 2322–2329. MIT Press.

Lissovoi, A.; Oliveto, P. S.; and Warwicker, J. A. 2019b. Simple hyper-heuristics control the neighbourhood size of randomised local search optimally for LeadingOnes. *Evolutionary Computation*. Early Access. doi:10.1162/evco_a_00258.

Özcan, E.; Bilgin, B.; and Korkmaz, E. E. 2008. A comprehensive analysis of hyper-heuristics. *Intelligent Data Analysis* 12(1):3–23.

Qian, C.; Tang, K.; and Zhou, Z.-H. 2016. Selection hyper-heuristics can provably be helpful in evolutionary multi-objective optimization. In *Proceedings of the International Conference on Parallel Problem Solving from Nature*, PPSN '16, 835–846. Springer.

Quick, R. J.; Rayward-Smith, V. J.; and Smith, G. D. 1998. Fitness distance correlation and ridge functions. In *Proceedings of the International Conference on Parallel Problem Solving from Nature*, PPSN '98, 77–86. Springer.

Witt, C. 2014. Fitness levels with tail bounds for the analysis of randomized search heuristics. *Information Processing Letters* 114(1-2):38–41.