2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Runtime Analysis of Selection Hyper-heuristics with Classical Learning Mechanisms

Fawaz Alanazi, Per Kristian Lehre

***Abstract*—The term *selection hyper-heuristics* refers to a randomised search technique used to solve computational problems by choosing and executing heuristics from a set of pre-defined low-level heuristic components. Selection hyper-heuristics have been successfully employed in many problem domains. Nevertheless, a theoretical foundation of these heuristics is largely missing. Gaining insight into the behaviour of selection hyper-heuristics is challenging due to the complexity and random design of these heuristics. This paper is one of the initial studies to analyse rigorously the runtime of selection hyper-heuristics with a number of the most commonly used learning mechanisms; namely, simple random, random gradient, greedy, and permutation. We derive the runtime of selection hyper-heuristic with these learning mechanisms not only on a classical example problem, but also on a general model of fitness landscapes. This in turn helps in understanding the behaviour of hyper-heuristics. Our results show that all the considered selections hyper-heuristics have roughly the same performance. This suggests that the learning mechanisms do not necessarily improve the performance of hyper-heuristics. A new learning mechanism that improves the performance of hyper-heuristic on our example problem is presented.**

I. Introduction

Hyper-heuristics are search strategies that for either selecting or generating new heuristics based on a set of pre-defined low-level heuristics. Unlike most other heuristic search techniques, hyper-heuristics are not problem-specific methods. This can be considered as one of the key features. Recently, selection hyper-heuristics have become popular for tackling some instances of NP-hard real-world problems [1]. A generic selection hyper-heuristic consists of two strategies; namely, *heuristic selection* or so-called *learning mechanism* and *move acceptance* strategy [2]. The former is used to select a heuristic from a fixed set of low-level heuristics (e.g. mutation heuristics, crossover heuristics) based on some probability distributions. The second strategy is used once the selected heuristic is applied on the candidate solution to decide whether the new solution is accepted or discarded. Several move acceptance methods were used within selection hyper-heuristics frameworks. For example, the *only improving* method [3] accepts the new solution if and only if it is better than the current candidate solution. This paper considers *only improving* move acceptance method only.

Various learning mechanisms have been proposed in the literature [4] [5]. However, there were no previous mathematical definitions of learning mechanisms. All learning mechanisms can be defined as follows:

Fawaz Alanazi and Per Kristian Lehre are with ASAP Research Group, The Department of Computer Science, The University of Nottingham, UK (email: {fza, pkl}@cs.nott.ac.uk).

Definition *1:* Let $\mathscr{X}$ be a finite search space and $f : \mathscr{X} \to \mathbb{R}$ be a cost function. Let m be the number of low-level heuristics, and $h_j^{(t)}$ be the selected heuristics in iteration t. Let $p(h_k^{(t+1)})$ be the selection probability of heuristic k in iteration $t+1$. A learning mechanism within a selection hyper-heuristic framework can be defined as a function ℓ:

$$\left((h_j^{(i)}, f(x_i))\right)_{i=1..t} \mapsto \left(p(h_k^{(t+1)})\right)_{k=1..m}$$

where $\sum_{k=1}^{m} p(h_k^{(t+1)}) = 1$

Learning mechanisms are given the information about the cost values of the solutions from the beginning until the current iteration t, as well as the low-level heuristics that were used in each iteration. Then they return probability distribution over the set of low-level heuristics. On the one hand, all learning mechanisms aim at selecting all low-level heuristics based on some probability distributions. On the other hand, different learning mechanisms have different strategies. For example, some learning mechanisms update the selection probability of low-level heuristics based on the available information about their performance in previous steps; whereas others choose heuristics with a fixed-selection probability regardless of their performances.

Despite the great success of selection hyper-heuristics in several problem domains, a theoretical analysis of such heuristics is largely ignored [6] [7]. Runtime analysis refers to theoretical studies that rigorously estimate the runtime and success probability of randomised search heuristics, where the success probability is the probability of a search heuristic to find an optimal solution within a specified time. One of the first runtime analysis of hyper-heuristics was presented in [6]. The expected runtime of simple hyper-heuristics was analysed. The authors conclude by that mixing low-level heuristics is efficient with the appropriate parameter settings. This study performs rigorous runtime analysis of selection hyper-heuristics with several learning mechanisms; namely, simple random, random gradient, greedy, and permutation. The motivation is not only to examine whether the learning schemes in these mechanisms can improve the performance of selection hyper-heuristics, but also to compare the runtime of selection hyper-heuristic with different characteristics on both a general scenario and a classical example problem.

A. Notation

The following notation is used in this paper. For $n > 0$, let n be the set of integers $[n] = \{1, 2, .., n\}$, usually n is the bit-string length. The notation X_t denotes the state of a solution in iteration t, e.g. in section III, it is the number of non-leading one bits in iteration t. Let x_{min} and x_{max} be the minimum and maximum value X_t can take,

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

respectively. Standard notation for the runtime analysis, e.g. big O notation, is used.

II. Preliminaries

A number of analytical techniques were investigated in the runtime analysis of randomised search heuristics. Drift analysis is one of the most popular among these techniques. It is centred on the analysis of a randomised search heuristic behaviour in a single step. This process is performed by measuring the progress of a function, a so-called *distance function*, which assigns each state of the search heuristic to a non-negative number reflecting the distance between that state and the optimal solution. A range of drift theorems have been presented in the literature (see for example [8] [9]). Very recently, a general drift theorem that can be considered to be a generalisation of most of the existing drift theorems is introduced in [10].

Theorem *1:* [10]
Let $(X_t)_{t\geq 0}$ be a stochastic process over some state space $S \subseteq \{0\} \cup [x_{min}, x_{max}]$, where $x_{min} > 0$. Let $h : [x_{min}, x_{max}] \to \mathbb{R}^+$ be an integrable function and define $g : 0 \cup [x_{min}, x_{max}] \to \mathbb{R}^{\geq 0}$ by $g(x) = \frac{x_{min}}{h(x_{min})} + \int_{x_{min}}^{x} \frac{1}{h(y)} dy$ for $x \geq x_{min}$ and $g(0) = 0$. Then, the following statements holds for the first hitting time $T = min\{t | X_t = 0\}$.

- If $\mathbb{E}[X_t - X_{t+1} | X_t \geq x_{min}] \geq h(X_t)$ and $\mathbb{E}[g(X_t) - g(X_{t+1}) | X_t \geq x_{min}] \geq \delta$, then:
$$\mathbb{E}[T|X_0] \leq \frac{g(X_0)}{\delta}.$$
- If $\mathbb{E}[X_t - X_{t+1} | X_t \geq x_{min}] \leq h(X_t)$ and $\mathbb{E}[g(X_t) - g(X_{t+1}) | X_t \geq x_{min}] \leq \alpha$, then:
$$\mathbb{E}[T|X_0] \geq \frac{g(X_0)}{\alpha}.$$

According to [10], the first statements of this theorem can be simplified as follows:

Corollary *1:* If $\mathbb{E}[X_t - X_{t+1} | X_t \geq x_{min}] \geq h(X_t)$ and $h'(x) \geq 0$, then $\delta \geq 1$ and hence: $\mathbb{E}[T|X_0] \leq g(X_0)$.

The artificial fitness levels technique was one of the first techniques developed to analyse the runtime of randomised search heuristics [11]. In this technique, the search space must be divided into a number of fitness-levels depending on the fitness function value of that particular fitness-level, where the solution quality in fitness-level j must be better than that in fitness-level i for all $i < j$. This implies that the optimal solution is always in the last fitness-level. The artificial fitness levels technique is defined precisely as follows [11].

Definition *2:* Let S be a finite search space, and $(S_1, S_2, .., S_n)$ be the fitness levels partition of a function $f : S \to \mathbb{R}$, then it must hold:

- $f(S_i) < f(S_j)$, for any fitness level $i < j$.
- $S_i \cap S_j = \phi$, for all i, j.
- The optimal solution is always in the last fitness level.

Theorem *2:* [11]
Let S be a finite search space, and $(S_1, .., S_n)$ be the fitness-levels partition of a function $f : S \to \mathbb{R}$. If p_i is a lower bound on the probability to leave fitness-level i towards a higher fitness-level, then the expected time to reach S_n is at most:
$$\mathbb{E}[T] \leq \sum_{i=1}^{n-1} \frac{1}{p_i}$$

III. Runtime analysis of Selection Hyper-heuristics

Generic selection hyper-heuristics work on low-level heuristics and search spaces. However, they monitor a single objective function representing the quality of the solution at hand. Algorithm 1 presents a pseudo-code of generic selection hyper-heuristics. This section shows the runtime analysis

Algorithm 1 Selection Hyper-heuristic

Let:
- S be a finite search space
- H be a set of low-level heuristics
- $f : S \to \mathbb{R}^+$ be a cost function

$s \sim Unif(S)$
while stopping conditions not satisfied **do**
  $h_j \leftarrow \ell(H)$ //a learning mechanism ℓ is used to choose a heuristic from H.
  $s' = h_j(s)$ //execute the selected heuristic
  if $f(s') > f(s)$ **then**
    $s = s'$
  end if
end while

of selection hyper-heuristics with various learning mechanisms. We consider both a general model of fitness landscapes as well as the specific test problem LeadingOnes, which is a well-known test problem in runtime analysis. LeadingOnes counts the number of consecutive, leading one-bits:
$$\textsc{LeadingOnes}(x) = \sum_{i=1}^{n} \prod_{j=1}^{i} x_j$$

We analyse the runtime of selection hyper-heuristic on LeadingOnes based on the following assumptions:

- It is assumed that all the bits in the initial bit-string are distributed uniformly at random. In order for the LeadingOnes function to be improved, at least the left-most 0-bit must be flipped and the leading one bits remain the same.
- Drift analysis is used as runtime analysis technique. We set $x_{min} = 2$, $X_0 = n - 1$, and let X_t be the number of non-leading one-bits in iteration t.
- We use two mutation operators as low-level heuristics:
  1) The first mutation operator (1OP) flips one randomly chosen bit position. The success probability of this operator at state X_t is $p(X_t) = \frac{1}{n}$, where n is the length of the bit-string.

2) The second mutation operator (2OP) flips two bits chosen uniformly at random. Its success probability is $q(X_t) = \frac{2X_t-2}{n^2}$.

Furthermore, we consider a general model of fitness landscapes that is suitable for any function that satisfies the conditions of the artificial fitness levels technique as described in section II. The following subsections present the expected runtime of selection hyper-heuristics on LEADINGONES and for all general functions.

A. Simple Random

Simple random selection hyper-heuristics choose a heuristic from a set of low-level heuristics uniformly at random in every iteration [3]. It does not use any available feedback; hence, each heuristic has an equal opportunity of being chosen in every iteration.

Lemma *1:* Algorithm 1 with simple random on LEADINGONES has drift

$$\frac{n+2X_t-2}{2n^2} \leq \mathbb{E}[X_t - X_{t+1} | X_t \geq x_{min}] \leq \frac{n+2X_t-2}{n^2}.$$

Proof: Let K be the number of consecutive 1-bits after the left-most 0-bit (so-called "free riders"). Since every bit after the left-most 0-bit is a 1-bit with probability $1/2$, the expected value of K is:

$$\begin{aligned}
\mathbb{E}[K \mid X_t] &= \sum_{i=1}^{X_t-1} i \cdot \left(\frac{1}{2}\right)^{i+1} \\
&= \frac{1}{2} \sum_{i=1}^{X_t-1} i \cdot \left(\frac{1}{2}\right)^{i} \\
&= \frac{1}{2} \left(\frac{(X_t-1)(1/2)^{X_t+1} - X_t(1/2)^{X_t} + (1/2)}{1/4} \right) \\
&= 1 - 2^{-X_t}(X_t+1).
\end{aligned}$$

Because $2 \leq X_t \leq n-1$, it holds that $0 \leq \mathbb{E}[K \mid X_t] \leq 1$, so, the expected progress in an improving step is at least 1 and at most 2. Simple random chooses a mutation operator uniformly at random in every iteration. The expected change in distance is bounded from below by

$$\mathbb{E}[X_t - X_{t+1} | X_t \geq x_{min}] \geq \frac{1}{2}(p(X_t) + q(X_t))$$

and from above by

$$\mathbb{E}[X_t - X_{t+1} | X_t \geq x_{min}] \leq p(X_t) + q(X_t)$$

where $p(X_t) = 1/n$, and $q(X_t) = 2(X_t-1)/n^2$. ∎

Theorem *3:* The expected runtime of Algorithm 1 with simple random, and $H = \{1OP, 2OP\}$, on LEADINGONES is:

$$\mathbb{E}[T|X_0] = n^2 \ln(3) + O(n).$$

Proof: From Lemma 1, it follows that

$$\mathbb{E}[X_t - X_{t+1} | X_t > x_{min}] \geq \frac{n+2X_t-2}{2n^2} =: h(X_t).$$

Since $h'(x) = 1/n^2 > 0$, the expected runtime of Algorithm 1 is by Theorem 1

$$\begin{aligned}
\mathbb{E}[T|X_0] &\leq \frac{x_{min}}{h(x_{min})} + \int_{x_{min}}^{X_0} \frac{1}{h(y)} dy \\
&\leq \frac{4n^2}{n+2} + \int_2^{n-1} \frac{2n^2}{n+2y-2} dy \\
&\leq n^2 \left(\frac{4}{n+2} + \ln\left(\frac{3n-4}{n+2} \right) \right).
\end{aligned}$$

The result now follows by noting that $\frac{4}{n+2} = O(1/n)$ and $\ln(\frac{3n-4}{n+2}) \leq \ln(3)$. ∎

Theorem *4:* The expected runtime of Algorithm 1 with simple random, and $H = \{1OP, 2OP\}$, on LEADINGONES is at least:

$$\mathbb{E}[T|X_0] \geq \frac{n^2}{6} \ln(3).$$

Proof: By Lemma 1,

$$\mathbb{E}[X_t - X_{t+1} | X_t > x_{min}] \leq \frac{n+2X_t}{n^2} =: h(X_t)$$

Hence,

$$\begin{aligned}
\mathbb{E}[g(X_t) - g(X_{t+1}) | X_t \geq x_{min}] &\leq \int_{X_{t+1}}^{X_t} \frac{1}{h(y)} dy \\
&\leq \frac{n^2}{2} \ln\left(\frac{n+2X_t}{n+2X_{t+1}} \right) \\
&\leq \frac{n^2}{2} \ln\left(1 + \frac{2(X_t - X_{t+1})}{n+2X_{t+1}} \right) \\
&\leq \frac{n^2}{2} \cdot \frac{2}{n} \cdot \mathbb{E}[X_t - X_{t+1} | X_t] \\
&\leq \frac{n(n+2X_t)}{n^2} \leq 3,
\end{aligned}$$

because the largest value X_t can take is $n-1$. The theorem now follows by the second statement in Theorem 1 with $\alpha = 3$.

$$\mathbb{E}[T|X_0] \geq \frac{g(X_0)}{3}$$

where,

$$\begin{aligned}
g(X_0) &= \frac{2n^2}{n+2} + \int_2^{n-1} \frac{n^2}{n+2y-2} dy \\
&\geq \frac{n^2}{2} \ln\left(\frac{3n-4}{n+2} \right) \\
&\geq \frac{n^2}{2} \ln(3).
\end{aligned}$$

∎

B. Random Gradient

Initially random gradient chooses a low-level heuristic uniformly at random. Then it applies the selected heuristic iteratively as long as an improvement in the objective function is obtained [5]. Although this mechanism tracks the historical performance of low-level heuristics, where it calls the heuristic that improved the solution in the previous step, the gathered information about the performance of these heuristics is lost as soon as the chosen heuristic fails to

improve the candidate solution. Therefore, we consider the drift of two steps instead of single step, e.g. $\mathbb{E}[X_t - X_{t+2}]$.

Lemma *2:* The expected two steps drift of Algorithm 1 with random gradient:
If $X_t \leq \frac{n}{2}+1$

$$\mathbb{E}[X_t - X_{t+2}|X_t \geq x_{min}] \geq \frac{n(n-2)+X_t(2n-1)-1}{2n^3}$$

and

$$\mathbb{E}[X_t - X_{t+2}|X_t \geq x_{min}] \leq \frac{6n+4(X_t-1)}{2n^2}$$

If $X_t > \frac{n}{2}+1$

$$\mathbb{E}[X_t - X_{t+2}|X_t \geq x_{min}] \geq \frac{3n+2X_t-2}{2n^2}$$

and

$$\mathbb{E}[X_t - X_{t+2}|X_t \geq x_{min}] \leq \frac{n(n-2)+X_t(2n-1)-1}{n^3}$$

Proof: Random gradient changes the probability distribution over the low-level heuristics based on the outcome of the previous iteration, thus we considered all the possible drift in two steps.
Case 1: An operator is chosen uniformly at random.

$$\begin{aligned}
&\mathbb{E}[X_t - X_{t+2}|X_t \geq x_{min}] \\
&= \left(1 - \frac{p(X_t)+q(X_t)}{2}\right)\left(\frac{p(X_t)+q(X_t)}{2}\right) \\
&\quad + \frac{p(X_t)(1-p(X_{t+1}))}{2} + \frac{q(X_t)(1-q(X_{t+1}))}{2} \\
&\quad + \frac{2p(X_t)p(X_{t+1})}{2} + \frac{2q(X_t)q(X_{t+1})}{2} \\
&= \frac{(n+2X_t-2)(4n^2-n-2X_t+2)}{4n^4} = h_1(X_t)
\end{aligned}$$

Case 2: The first operator is chosen with probability one.

$$\begin{aligned}
&\mathbb{E}[X_t - X_{t+2}|X_t \geq x_{min}] = p(X_t)(1-p(X_{t+1})) \\
&+ (1-p(X_t))\frac{p(X_t)+q(X_t)}{2} + 2p(X_t)p(X_{t+1}) \\
&= \frac{3n+2X_t-2}{2n^2} = h_2(X_t)
\end{aligned}$$

Case 3: The second operator is chosen with probability one.

$$\begin{aligned}
&\mathbb{E}[X_t - X_{t+2}|X_t \geq x_{min}] = q(X_t)(1-q(X_{t+1})) \\
&+ (1-q(X_t))\frac{p(X_t)+q(X_t)}{2} + 2q(X_t)q(X_{t+1}) \\
&= \frac{n(n-2)+X_t(2n-1)-1}{2n^3} = h_3(X_t)
\end{aligned}$$

The expected progress in an improving step is at least 1 and at most 2 as proved in Lemma 1, then $\mathbb{E}[X_t - X_{t+2}|X_t \geq x_{min}]$ if $X_t \leq \frac{n}{2}+1$:

$$h_3(X_t) \leq \mathbb{E}[X_t - X_{t+2}|X_t \geq x_{min}] \leq 2h_2(X_t)$$

Otherwise,

$$h_2(X_t) \leq \mathbb{E}[X_t - X_{t+2}|X_t \geq x_{min}] \leq 2h_3(X_t)$$

■

Theorem *5:* The expected optimisation time of random gradient selection hyper-heuristic, with $H = \{1OP, 2OP\}$, on LEADINGONES is at most:

$$\mathbb{E}[T|X_0] = 2n^2(\ln(5/2)+o(1))$$

Proof: As long as $X_t \leq \frac{n+2}{2}$ the expected drift is:

$$\begin{aligned}
&\mathbb{E}[X_t - X_{t+2}|X_t \geq x_{min}] \\
&\qquad \geq \frac{n(n-2)+X_t(2n-1)-1}{2n^3} = h_1(X_t)
\end{aligned}$$

Otherwise, the expected drift is at least:

$$\mathbb{E}[X_t - X_{t+2}|X_t \geq x_{min}] \geq \frac{3n+2X_t-2}{2n^2} = h_2(X_t)$$

Since $h_1'(x)$ and $h_2'(x)$ are larger than zero, the expected runtime of Algorithm 1 is by Theorem 1

$$\begin{aligned}
\mathbb{E}[T|X_0] &\leq \frac{x_{min}}{h_1(x_{min})} + \int_{x_{min}}^{\frac{n}{2}+1}\frac{1}{h_1(y)}dy + \int_{\frac{n}{2}+2}^{n-1}\frac{1}{h_2(y)}dy \\
&\leq \frac{4n^2}{n+2} + \int_2^{\frac{n}{2}+1}\frac{2n^3}{n(n-2)+y(2n-1)-1}dy \\
&+ \int_{\frac{n}{2}+2}^{n-1}\frac{2n^2}{3n+2y-2}dy \\
&\leq n^2\left(\frac{4}{n+2} + \ln(\frac{5n-4}{4n+2}) + \ln(\frac{-4n^2+n+4}{-2n^2-4n+6})\right)
\end{aligned}$$

where,

$$\begin{aligned}
\frac{-4n^2+n+4}{-2n^2-4n+6} &= \frac{4n^2-n-4}{2n^2+4n-6} \\
&= \frac{2n^2+4n-6}{2n^2+4n-6} + \frac{2n^2-5n+2}{2n^2+4n-6} \\
&= 1 + \frac{2n^2-5n+2}{2n^2+4n-6} < 2
\end{aligned}$$

and since $\ln(\frac{5n-4}{4n+2}) \leq \ln(\frac{5}{4})$, then

$$\mathbb{E}[T|X_0] = n^2(\ln(2)+\ln(5/4)+o(1))$$

Since we used the drift of two steps, then the expected runtime of Algorithm 1:

$$\mathbb{E}[T|X_0] = 2n^2(\ln(5/2)+o(1))$$

■

Theorem *6:* The expected runtime of random gradient selection hyper-heuristic, with $H = \{OP1, OP2\}$, on LEADINGONES is at least:

$$\mathbb{E}[T|X_0] \geq \frac{n^2}{9}(4+3\cdot\ln(10/3))$$

Proof: Let $X_t = n-1$, then the expected drift of Algorithm 1 is at most:

$$\mathbb{E}[X_t - X_{t+2}|X_t \geq x_{min}] \leq \frac{n(n-2)+X_t(2n-1)-1}{n^3}$$

$$\mathbb{E}[g(X_t) - g(X_t t+1) | X_t \geq x_{min}] \leq$$
$$\int_{X_{t+1}}^{X_t} \frac{2n^3}{2n(n-2)+2y(2n-1)-2} dy$$
$$\leq \frac{n^2}{2}\left(\ln(\frac{n^2+2n(X_t-1)}{n^2+2n(X_{t+1}-1)})\right)$$
$$\leq \frac{n^2}{2}\left(\ln(\frac{n+2X_t}{n+2X_{t+1}})\right)$$
$$\leq \frac{n^2}{2}\left(\frac{2}{n}\cdot\mathbb{E}[X_t - X_{t+1}|X_t]\right)$$
$$\leq n\cdot\frac{6n^2-4n}{2n^3} = 3$$

Then the expected runtime of this algorithm on LEADINGONES is at least:

$$\mathbb{E}[T|X_0] \geq 2(\frac{g(X_0)}{3})$$
$$\geq \frac{2}{3}\cdot\Big(\frac{2n}{3} + \int_2^{\frac{n}{2}+1} \frac{2n^2}{6n+4(y-1)} dy$$
$$+ \int_{\frac{n}{2}+2}^{n-1} \frac{2n^3}{2n(n-2)+2y(2n-1)-2} dy\Big)$$
$$\geq \frac{n^2}{9}\left(4+3(\ln(\frac{5n+2}{3n+2}) + ln(\frac{6n^2-10n}{4n^2+3n-6}))\right)$$

where,

$$\frac{6n^2-10n}{4n^2+3n-6} = \frac{4n^2-3n-6}{4n^2+3n-6} + \frac{2n^2-7n+6}{4n^2+3n-6} \leq 2$$

then,

$$\mathbb{E}[T|X_0] \geq \frac{n^2}{9}(4+3(\ln(5/3)+ln(2))$$
$$\geq \frac{n^2}{9}(4+3\cdot\ln(10/3))$$

■

We now consider general model of fitness landscapes for any function as described in section II is also considered. Lehre and Ozcan in [6] presented the runtime of simple random hyper-heuristic in this scenario (see Theorem 2 in [6]). The following theorem presented the expected runtime of Algorithm 1 by using random gradient as learning mechanism for any function that satisfies the artificial fitness levels conditions.

Theorem *7:* Let S be a finite search space divided into a number of fitness-levels $(S_1, S_2, .., S_n)$. Given any objective function $f : S \to \mathbb{R}^+$. Let m be the number of low-level heuristics and $p_j^{(k)}$ be the smallest success probability of the j^{th} heuristic at fitness-level k. If $\forall j \in m\ p_j^{(a)} = p_j^{(b)}$ for all fitness level $a, b \in n$, then the expected runtime of random gradient selection hyper-heuristic is:

$$\mathbb{E}[T] \leq (n-1)\left(1 + \frac{m}{\sum_{j=1}^m p_j}(1 - \frac{\sum_{j=1}^m p_j^2}{\sum_{j=1}^m p_j})\right)$$

Proof: We prove by induction that for all $k \in [1,n]$:

$$\Pr(S_j^{(k)} = 1) = \frac{p_j^{(k)}}{\sum_{i=1}^m p_i^{(k)}} \quad (1)$$

where $\Pr(S_j^{(k)} = 1)$ is the success probability of the j^{th} heuristic in fitness level k.
base case: when $k = 1$, Since random gradient chooses a heuristic uniformly at randomly at the beginning, the success probability of the j^{th} heuristic in fitness level $S^{(1)}$ is:

$$\Pr(S_j^{(2)} = 1) = \frac{p_j^{(1)}}{\sum_{i=1}^m p_i^{(1)}} \quad (2)$$

Induction step: Let R^k be a random variable that takes one when the selected heuristic fails to improve the candidate solution, and zero otherwise. Suppose $k = z$, Then by the law of total probability:

$$\Pr(S_j^{(z+1)} = 1)$$
$$= R^{(z)}(\frac{p_j^{(z)}}{\sum_{i=1}^m p_i^{(z)}}) + (1-R^{(z)})\Pr(S_j^{(z)} = 1)$$
$$= R^{(z)}(\frac{p_j^{(z)}}{\sum_{i=1}^m p_i^{(z)}}) + (1-R^{(z)})(\frac{p_j^{(z)}}{\sum_{i=1}^m p_k^{(z)}})$$
$$= \frac{p_j^{(z)}}{\sum_{i=1}^m p_i^{(z)}}(R^{(z)} + 1 - R^{(z)})$$
$$= \frac{p_j^{(z)}}{\sum_{i=1}^m p_i^{(z)}} \quad (3)$$

By induction, (1) is true for all $k \in [1,n]$. Thus by the law of the total expectation:

$$\mathbb{E}[R^{(k)}] = \sum_{j=1}^m \Pr(S_j^{(k)} = 1)\mathbb{E}[R^{(k)} \mid S_j^{(k)} = 1]$$
$$= \sum_{j=1}^m \frac{p_j^{(k)}}{\sum_{i=1}^m p_i^{(k)}}(1 - p_j^{(k)})$$
$$= \frac{1}{\sum_{i=1}^m p_i^{(k)}}\Big(\sum_{j=1}^m p_j^{(k)}(1-p_j^{(k)})\Big)$$

Once the selected heuristic fails to improve the candidate solution, random gradient chooses a heuristic uniformly at random. The expected waiting time to reach a higher fitness level by selecting a heuristic uniformly at random is:

$$\mathbb{E}[T_k] = \frac{1}{\frac{1}{m}\sum_{j=1}^m p_j^{(k)}}$$

Since this mechanism needs one iteration to examine the chosen heuristic, the expected runtime of random gradient hyper-heuristic is at most:

$$\mathbb{E}[T_{Alg2}] \leq \sum_{k=1}^{n-1} 1 + \mathbb{E}[R^{(k)}](\frac{m}{\sum_{j=1}^m p_j^{(k)}})$$
$$\leq (n-1)\Big(1 + \frac{m}{\sum_{j=1}^m p_j} - \frac{m\sum_{j=1}^m p_j^2}{(\sum_{j=1}^m)^2}\Big)$$

■

C. Greedy

The greedy learning mechanism applies all low-level heuristics to the same candidate solution then deterministically chooses the heuristic that achieves the best change in the objective function [3]. This mechanism learns nothing from the historical performance of low-level heuristics, where the probability distribution over the heuristics space is changed based on the performance of these heuristics on the current search stage.

Lemma *3:* Greedy selection hyper-heuristic with $H = \{1OP, 2OP\}$ has drift:

$$\mathbb{E}[X_t - X_{t+1} | X_t \geq x_{min}] \geq \frac{2X_t(n-2)+n(n-2)+2}{n^3}$$

and

$$\mathbb{E}[X_t - X_{t+1} | X_t \geq x_{min}] \leq \frac{4X_t(n-2)+2n(n-2)+4}{n^3}$$

Proof: Greedy applies both mutation operators to the same candidate solution in every step; hence, the success probability is at least:

$$\begin{aligned}\mathbb{E}[X_t - X_{t+1} | X_t \geq x_{min}] &\geq 1-(1-p(X_t))(1-q(X_t)) \\ &\geq \frac{2X_t(n-2)+n(n-2)+2}{n^3}\end{aligned}$$

The second statement can be proved in symmetrical way to the first statement taking into account the lower and upper expected progress in an improving step. ∎

Theorem *8:* The expected runtime of Algorithm 1 with greedy, and $H = \{1OP, 2OP\}$, on LEADINGONES is at most:

$$\mathbb{E}[T|X_0] = n^2(\ln(3)+o(1))$$

Proof:

$$\mathbb{E}[X_t - X_{t+1}] \geq \frac{2X_t(n-2)+n(n-2)+2}{n^3}$$

The expected runtime of Algorithm 1 is by Theorem 1

$$\begin{aligned}\mathbb{E}[T|X_0] &\leq \frac{2n^2}{n+2} + \int_2^{n-1} \frac{n^3}{2y(n-2)+n(n-2)+2} dy \\ &\leq \frac{n^2}{2}\left(\frac{4}{n+2} + \ln(\frac{3n^2-8n+6}{n^2+2n-6})\right)\end{aligned}$$

where,

$$\begin{aligned}\frac{3n^2-8n+6}{n^2+2n-6} &= \frac{n^2+2n-6}{n^2+2n-6} + \frac{2n^2-10n+12}{n^2+2n-6} \\ &= 1+2\cdot(\frac{n^2-5n+6}{n^2+2n-6}) \\ &\leq 1+2\cdot(1) = 3\end{aligned}$$

Since greedy applies both operators in every step, it holds:

$$\begin{aligned}\mathbb{E}[T|X_0] &\leq 2\left(\frac{n^2}{2}(\frac{4}{n+2}+\ln(3))\right) \\ &\leq n^2(\ln(3)+o(1))\end{aligned}$$

∎

Theorem *9:* The expected runtime of Algorithm 1 with greedy learning mechanism, and $H = \{1OP, 2OP\}$, on LEADINGONES is at least:

$$\mathbb{E}[T|X_0] \geq \frac{n^2}{6}(\ln(3)+o(1))$$

Proof: From Lemma 3, the expected drift of this algorithm is at most $\frac{4X_t(n-2)+2n(n-2)+4}{n^3}$.

$$\begin{aligned}&\mathbb{E}[g(X_t) - g(X_{t+1}) | X_t \geq x_{min}] \leq \\ &\int_{X_{t+1}}^{X_t} \frac{n^3}{4y(n-2)+2n(n-2)+4} dy \\ &\leq \frac{n^2}{4}\left(\ln(\frac{4X_t(n-2)+2n(n-2)+4}{4X_{t+1}(n-2)+2n(n-2)+4})\right) \\ &\leq \frac{n^2}{4}\left(\ln(\frac{4X_t\cdot n+2n^2}{4X_{t+1}\cdot n+2n^2})\right) \\ &\leq \frac{n^2}{4}\left(\ln(1+\frac{4n(X_t-X_{t+1})}{2n^2})\right) \\ &\leq \frac{n^2}{4}\left(\frac{4n}{2n^2}\cdot\frac{4X_t(n-2)+n(n-2)+4}{n^3}\right)\end{aligned}$$

where $X_t \leq n-1$, then:

$$\mathbb{E}[g(X_t) - g(X_{t+1}) | X_t \geq x_{min}] \leq \frac{n}{2}\cdot\frac{6n^2-10n}{n^3} = 3$$

The expected runtime of Algorithm 1 on LEADINGONES is, therefore, at least:

$$\begin{aligned}\mathbb{E}[T|X_0] &\geq 2\left(\frac{g(X_0)}{3}\right) \\ &\geq \frac{2}{3}\left(\frac{n^2}{n+2} + \int_2^{n-1} \frac{n^3}{4y(n-2)+2n(n-2)+4} dy\right) \\ &\geq \frac{2}{3}\left(\frac{n^2}{4}(\frac{8}{n+2} + \ln(\frac{3n^2-8n+6}{n^2+2n-6}))\right)\end{aligned}$$

Since $\frac{3n^2-8n+6}{n^2+2n-6} \leq 3$ as proved in Theorem 8, it follows that:

$$\begin{aligned}\mathbb{E}[T|X_0] &\geq \frac{2}{3}\left(\frac{n^2}{4}(\ln(3)+o(1))\right) \\ &\geq \frac{n^2}{6}(\ln(3)+o(1))\end{aligned}$$

∎

We now drive the runtime of greedy selection hyper-heuristic on any function that satisfies the artificial fitness levels technique conditions.

Theorem *10:* Let S be a finite search space divided into a number of fitness-levels $(S_1, S_2, .., S_n)$. Given any objective function $f : S \to \mathbb{R}^+$. Let m be the number of low-level heuristics and $p_j^{(k)}$ be the smallest success probability of the j^{th} heuristic at fitness-level k. Then the expected runtime of greedy selection hyper-heuristic is:

$$\mathbb{E}[T] \leq m(n-1) + \sum_{k=1}^{n-1} \frac{1}{1-\prod_{j=1}^{m}(1-p_k^{(j)})}$$

Proof: Greedy requires m iterations to apply all low-level heuristics to the same candidate solution. Employ Theorem 2 with success probability $p^{(k)}$, where $p^{(k)}$ $\forall$ fitness-level $k \in S$
is at least:

$$p^{(k)} \geq 1 - \left(\prod_{j=1}^{m}(1-p_j^{(k)})\right)$$

Then the expected time to leave fitness level k towards a higher fitness level is at most:

$$\mathbb{E}[T_k] = m + \frac{1}{1-\left(\prod_{j=1}^{m}(1-p_j^{(k)})\right)}$$

Hence,

$$\mathbb{E}[T] \leq \sum_{k=1}^{n-1} m + \frac{1}{1-\left(\prod_{j=1}^{m}(1-p_j^{(k)})\right)}$$

■

D. Permutation

The permutation mechanism generates a random order of all low-level heuristics, then in every iteration the next heuristic in the prepared order is applied [5].

Lemma *4:* The expected drift of permutation selection hyper-heuristic on LEADINGONES is:

$$\mathbb{E}[X_t - X_{t+2} | X_t \geq x_{min}] \geq \frac{n+2X_t-2}{n^2}$$

and

$$\mathbb{E}[X_t - X_{t+2} | X_t \geq x_{min}] \leq \frac{2n+4(X_t-1)}{n^2}$$

Proof: We consider the drift of two steps for more accurate runtime analysis:
Case 1: If the first mutation operator is in the top of the list.

$$\begin{aligned}&\mathbb{E}[X_t - X_{t+2} | X_t \geq x_{min}] = p(X_t)(1-q(X_{t+1}))\\&+(1-p(X_t))q(X_t)+2p(X_t)q(X_{t+1})\\&=\frac{n+2X_t-2}{n^2}\end{aligned}$$

Case 2: If the second mutation operator is in the top of the list.

$$\begin{aligned}&\mathbb{E}[X_t - X_{t+2} | X_t \geq x_{min}] = q(X_t)(1-p(X_{t+1}))\\&+(1-q(X_t))p(X_t)+2q(X_t)p(X_{t+1})\\&=\frac{n+2X_t-2}{n^2}\end{aligned}$$

It can be seen that the drift in both cases are the same, then the expected drift of $\mathbb{E}[X_t - X_{t+2} | X_t \geq x_{min}]$ is at least $(\frac{n+2X_t-2}{n^2})$ and at most $(\frac{2n+4(X_t-1)}{n^2})$. ■

Theorem *11:* The expected runtime of permutation hyper-heuristic, with $H = \{1OP, 2OP\}$, on LEADINGONES is at most:

$$\mathbb{E}[T|X_0] = n^2(\ln(3)+o(1))$$

Proof:

$$\mathbb{E}[X_t - X_{t+2} | X_t \geq x_{min}] \leq \frac{n+2X_t-2}{n^2} = h(X_t)$$

Since we used the drift of two steps,

$$\begin{aligned}\mathbb{E}[T|X_0] &\leq 2\left(\frac{2n^2}{n+2}+\int_2^{n-1}\frac{n^2}{n+2y-2}dy\right)\\&\leq n^2\left(\frac{4}{n+2}+\ln(\frac{3n-4}{n+2})\right)\end{aligned}$$

Since $\frac{3n-4}{n+2} \leq 3$, $\mathbb{E}[T|X_0] \leq n^2(\ln(3)+o(1))$ ■

Theorem *12:* The expected runtime of permutation hyper-heuristic, with $H = \{1OP, 2OP\}$, on LEADINGONES is at least:

$$\mathbb{E}[T|X_0] \geq \frac{n^2}{6}(\ln(3)+o(1))$$

Proof: From Lemma 4:

$$\mathbb{E}[X_t - X_{t+2} | X_t \geq x_{min}] \leq \frac{2n+4(X_t-1)}{n^2} = h(X_t)$$

$$\begin{aligned}&\mathbb{E}[g(X_t)-g(X_{t+1})|X_t \geq x_{min}] \leq \int_{X_{t+1}}^{X_t}\frac{n^2}{2n+4(y-1)}dy\\&\leq \frac{n^2}{4}\left(\ln(\frac{n+2X_t-2}{n+2X_{t+1}-2})\right)\\&\leq \frac{n^2}{4}\left(\ln(1+\frac{2(X_t-X_{t+1})}{n^2})\right)\\&\leq \frac{n^2}{4}\left(\frac{2}{n}.\frac{2n+4X_t-4}{n^2}\right)\end{aligned}$$

Where $X_t \leq n-1$,

$$\mathbb{E}[g(X_t)-g(X_{t+1})|X_t \geq x_{min}] \leq \frac{n}{2}.\frac{6n}{n^2} = 3$$

Since we used the drift of two steps, then:

$$\begin{aligned}\mathbb{E}[T|X_0] &\geq \frac{2}{3}\left(\frac{n^2}{n+2}+\int_2^{n-1}\frac{n^2}{2n+4(y-1)}dy\right)\\&\geq \frac{2}{3}\left(\frac{n^2}{4}(\frac{4}{n+2}+\ln(\frac{3n-4}{n+2}))\right)\\&\geq \frac{n^2}{6}(\ln(3)+o(1))\end{aligned}$$

■

IV. EXPERIMENTS

In addition to the theoretical analysis, a group of experiments were conducted. All the presented hyper-heuristics are used to optimise LEADINGONES function with settings as specified in section III. A run is continued until the optimal solution is found (1^n bit-string). Various lengths of the bit-string are considered, and for each length, 1000 instances are generated randomly. In each instance, these algorithms are run simultaneously on the same initial solution. The experimental results are summarised in the following charts (see Figure 1).

It can be seen that the performance of hyper-heuristics with the presented learning mechanisms is almost the same as

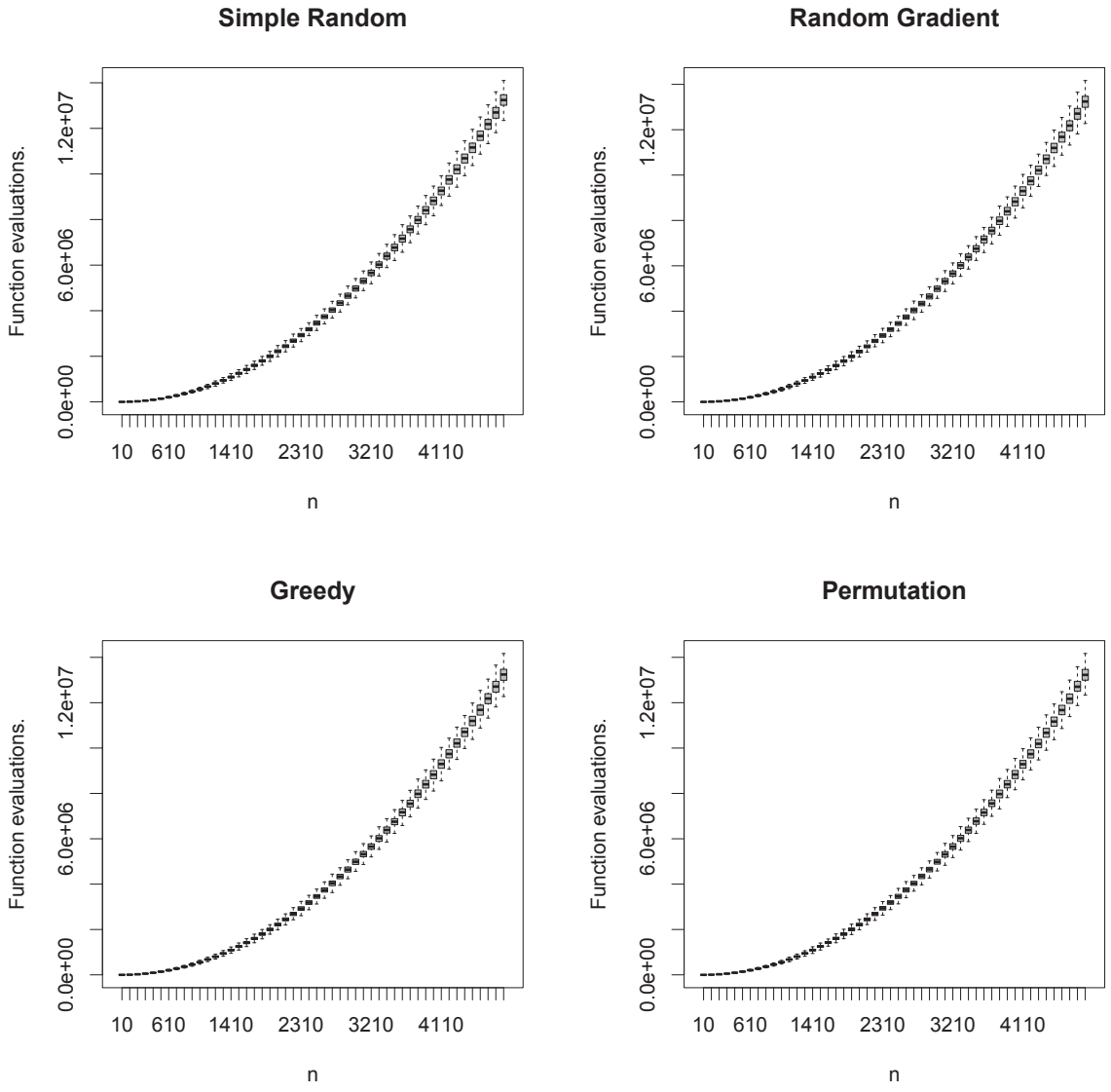

Fig. 1: Runtime of hyper-heuristics with simple random, random gradient, greedy, and permutation mechanisms on LEADINGONES.

shown in Figure 1. The learning schemes in random gradient and greedy mechanisms do not improve the performance of hyper-heuristic on our example problem compared to the other mechanisms that do not learn. The success probability of typical low-level heuristics is very small. Therefore, learning mechanisms that have learning schemes approximately choose low-level heuristics uniformly at random.

V. DISCUSSION

Our experimental and theoretical results have shown that the performance of selection hyper-heuristics with these learning mechanisms is almost the same in the specified settings. Table 1 summaries the theoretical runtime analysis results. It can be seen from the theoretical analysis that the first mutation operator $(1OP)$ is preferable if the distance (X_t) is less than or equal to $(\frac{n}{2}+1)$; otherwise, the second operator $(2OP)$ is superior. This, in fact, suggests a very important finding that the performance of low-level heuristics vary from one search stage to another. In addition, our theoretical analysis have also shown that the success probability of the used low-level heuristics, as the typical low-level heuristics, is very small. Consequently, we developed a new learning mechanism called improved random gradient. Our new learning mechanism is similar to random gradient except that it applies the successful heuristic iteratively for a specified number of iterations regardless the success of the selected heuristic. In our experiment, we set the time for improved random gradient to be from the beginning until $(X_t = \frac{n}{2}+1)$, and then until the selection hyper-heuristic find

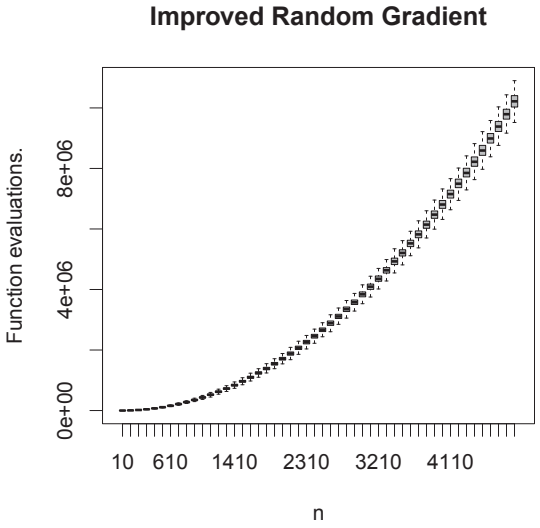

Fig. 2: Runtime of hyper-heuristic with the new learning mechanism on LEADINGONES.

Mechanism	Upper Bound	Lower Bound
Simple Random	$n^2\ln(3)+O(n)$	$\frac{n^2}{6}\ln(3)$
Random Gradient	$2n^2(\ln(5/2)+o(1))$	$\frac{n^2}{9}(4+3\cdot\ln(10/3))$
Greedy	$n^2(\ln(3)+o(1))$	$\frac{n^2}{6}(\ln(3)+o(1))$
Permutation	$n^2(\ln(3)+o(1))$	$\frac{n^2}{6}(\ln(3)+o(1))$

TABLE I: Runtime of Hyper-heuristics with Various Learning Mechanisms on LEADINGONES

the optimal solution. Our results showed that hyper-heuristics with improved random gradient outperformed all the other learning mechanisms on LEADINGONES as presented in Figure 2. Furthermore, this also led us to a very interesting finding that the performance of selection hyper-heuristics is improved with the right probability distribution over the low-level heuristics search space. The success probability of low-level heuristics is a very important factor that affect the performance of selection hyper-heuristics. It can also be deduced that learning schemes does not improve the performance of selection hyper-heuristics if the low-level heuristics have the same performance.

VI. CONCLUSION

This study is one of the initial studies on the runtime analysis of selection hyper-heuristics. The runtime of a generic selection hyper-heuristic with different learning mechanisms was investigated. The results have shown that the learning schemes in the presented learning mechanisms do not improve the performance of selection hyper-heuristics with the specified settings. A new learning mechanism that improved the performance of selection hyper-heuristic on our example problem is introduced. It can also be seen that specifying the most appropriate probability distribution over the heuristics space improve the performance of selection hyper-heuristics significantly. To conclude, hyper-heuristic is an area in which the theoretical background is still quite unexplored. For future work, a rigorous analysis of hyper-heuristics on more complex problems and learning mechanisms should be considered.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the British Engineering and Physical Science Research Council (EPSRC) grant no EP/F033214/1 (LANCS).

REFERENCES

[1] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and R. Qu, "Hyper-heuristics: A survey of the state of the art," *Journal of the Operational Research Society*, 2010.

[2] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. R. Woodward, "A classification of hyper-heuristic approaches," in *Handbook of Metaheuristics*. Springer, 2010, pp. 449–468.

[3] P. Cowling, G. Kendall, and E. Soubeiga, "A hyperheuristic approach to scheduling a sales summit," in *Practice and Theory of Automated Timetabling III*. Springer, 2001, pp. 176–190.

[4] E. Özcan, M. Misir, G. Ochoa, and E. K. Burke, "A reinforcement learning-great-deluge hyper-heuristic for examination timetabling," *International Journal of Applied Metaheuristic Computing (IJAMC)*, vol. 1, no. 1, pp. 39–59, 2010.

[5] P. Cowling, G. Kendall, and E. Soubeiga, "Hyperheuristics: A tool for rapid prototyping in scheduling and optimisation," in *Applications of Evolutionary Computing*. Springer, 2002, pp. 1–10.

[6] P. K. Lehre and E. Özcan, "A runtime analysis of simple hyper-heuristics: to mix or not to mix operators," in *Proceedings of the twelfth workshop on Foundations of genetic algorithms XII*. ACM, 2013, pp. 97–104.

[7] J. He, W. Hou, H. Dong, and F. He, "Mixed strategy may outperform pure strategy," *arXiv preprint arXiv:1303.3154*, 2013.

[8] P. K. Lehre, "Negative drift in populations," in *Parallel Problem Solving from Nature, PPSN XI*. Springer, 2010, pp. 244–253.

[9] J. E. Rowe and D. Sudholt, "The choice of the offspring population size in the $(1, \lambda)$ ea," in *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference*. ACM, 2012, pp. 1349–1356.

[10] P. K. Lehre and C. Witt, "General drift analysis with tail bounds," *arXiv preprint arXiv:1307.2559*, 2013.

[11] I. Wegener, *Theoretical aspects of evolutionary algorithms*. Springer, 2001.