Selection Hyper-heuristics Can Provably be Helpful in Evolutionary Multi-objective Optimization*

Chao Qian[1,2], Ke Tang[1], and Zhi-Hua Zhou[2]

[1]UBRI, School of Computer Science and Technology,
University of Science and Technology of China, Hefei 230027, China
[2]National Key Laboratory for Novel Software Technology,
Nanjing University, Nanjing 210023, China
{chaoqian,ketang}@ustc.edu.cn, zhouzh@nju.edu.cn

Abstract Selection hyper-heuristics are automated methodologies for selecting existing low-level heuristics to solve hard computational problems. They have been found very useful for evolutionary algorithms when solving both single and multi-objective real-world optimization problems. Previous work mainly focuses on empirical study, while theoretical study, particularly in multi-objective optimization, is largely insufficient. In this paper, we use three main components of multi-objective evolutionary algorithms (selection mechanisms, mutation operators, acceptance strategies) as low-level heuristics, respectively, and prove that using heuristic selection (i.e., mixing low-level heuristics) can be exponentially faster than using only one low-level heuristic. Our result provides theoretical support for multi-objective selection hyper-heuristics, and might be helpful for designing efficient heuristic selection methods in practice.

1 Introduction

Hyper-heuristics are automated methodologies for selecting or generating heuristics to solve hard computational problems [4]. There are two main hyper-heuristic categories: heuristic selection and heuristic generation. This paper focuses on the former type. Given a set of low-level heuristics, a heuristic selection method chooses an appropriate one to be applied at each decision point.

Selection hyper-heuristics have been widely and successfully applied for evolutionary algorithms (EAs) solving single-objective optimization problems such as personnel scheduling, packing, vehicle routing, etc [3]. After that, they start to emerge in evolutionary multi-objective optimization. Burke et al. [5] first proposed a multi-objective hyper-heuristic approach based on tabu search for the space allocation and timetabling problems. McClymont and Keedwell [17] developed a Markov chain based hyper-heuristic method for designing water distribution network. By using NSGAII, SPEA2 and MOGA as low-level heuristics, Maashi et al. [16] designed a choice function based hyper-heuristic to solve the vehicle crashworthiness design problem. Selec-

* This work was supported by the NSFC (61333014, 61329302), the Fundamental Research Funds for the Central Universities (WK2150110002), and the Collaborative Innovation Center of Novel Software Technology and Industrialization.

tion hyper-heuristics also achieved successes in other real-world multi-objective optimization problems, e.g., the 2D guillotine strip packing problem [18] and the integration and test order problem in software engineering [10].

Most previous work focuses on empirical study. Meanwhile, theoretical analysis, particularly running time analysis, is important for enhancing our understanding and designing efficient hyper-heuristics, as Burke et al. stated in [3]. However, the running time analysis on selection hyper-heuristics is difficult due to their complexity and randomness, and few results have been reported. By using mutation operators as low-level heuristics, He et al. [11] first gave some conditions under which the asymptotic hitting time of the (1+1)-EA (a simple EA) with a mixed strategy is not larger than that with any pure strategy. Note that a mixed strategy (which corresponds to heuristic selection) chooses one low-level heuristic according to some distribution each time, while a pure strategy uses only one fixed low-level heuristic. Their result was then extended to the expected running time measure and to population-based EAs in [12]. Lehre and Özcan [15] later gave concrete evidence that mixing two specific mutation operators is more efficient than using only one operator for the (1+1)-EA solving the GapPath function. They also proved the benefit of mixing acceptance strategies for the (1+1)-EA solving the RR_k function. In [19], by mixing global and local mutation operators, the (1+1)-EA was proved to be a polynomial time approximation algorithm for the NP-hard single machine scheduling problem. In [6], the (1+1)-EA mixing two specific mutation operators was shown to be able to solve the easiest function for each mutation operator efficiently. The above studies investigate whether selection hyper-heuristics can bring an improvement on the performance. Alanazi and Lehre [1] also compared different heuristic selection methods (i.e., mixed strategies with different distributions), and proved their similar performance for the (1+1)-EA solving the LeadingOnes function.

All of the above-mentioned studies consider single-objective optimization. To the best of our knowledge, there has been no theoretical work supporting the effectiveness of selection hyper-heuristics in multi-objective optimization. In this paper, we prove that using heuristic selection can speed up evolutionary multi-objective optimization exponentially via rigorous running time analysis. The widely used multi-objective EA GSEMO in previous theoretical analyses [8,14,21] is employed. It repeats three steps: choosing a solution by some selection mechanism, reproducing a new solution by mutation, and updating the population by some acceptance strategy. This paper considers the three main components of GSEMO, i.e., selection mechanism, mutation operator and acceptance strategy, as the low-level heuristic, respectively. For each kind of low-level heuristic, we give a bi-objective pseudo-Boolean function, and prove that the expected running time of GSEMO with a mixed strategy is polynomial while GSEMO with a pure strategy needs at least exponential running time. The analysis also shows that the helpfulness of selection hyper-heuristics is because the strengths of one heuristic can compensate for the weaknesses of another. For mixing acceptance strategies, we also empirically compare the running time of GSEMO with different mixed strategies, and the results imply the importance of choosing a proper heuristic selection method.

The rest of this paper is organized as follows. Section 2 introduces some preliminaries. The helpfulness of mixing selection mechanisms, mutation operators and acceptance strategies is then theoretically analyzed. Finally, we conclude the paper.

2 Preliminaries

Multi-objective optimization requires to simultaneously optimize two or more objective functions, as shown in Definition 1. Note that maximization is considered in this paper. The objectives are usually conflicted, and thus there is no canonical complete order on the solution space $\mathcal{X}$. The comparison between solutions relies on the *domination* relationship, as presented in Definition 2. A solution is *Pareto optimal* if there is no other solution in $\mathcal{X}$ that dominates it. The set of objective vectors of all the Pareto optimal solutions constitutes the *Pareto front*. The goal of multi-objective optimization is to find the Pareto front, that is, to find at least one corresponding solution for each element in the Pareto front. In this paper, we consider the Boolean space, i.e., $\mathcal{X} = \{0, 1\}^n$.

Definition 1 (Multi-Objective Optimization). *Given a feasible solution space $\mathcal{X}$ and objective functions $f_1, \ldots, f_m$, multi-objective optimization can be formulated as*

$$\max_{\boldsymbol{x} \in \mathcal{X}} \ \big(f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), ..., f_m(\boldsymbol{x})\big).$$

Definition 2 (Domination). *Let $\boldsymbol{f} = (f_1, f_2, \ldots, f_m) : \mathcal{X} \to \mathbb{R}^m$ be the objective vector. For two solutions $\boldsymbol{x}$ and $\boldsymbol{x}' \in \mathcal{X}$:*

1. $\boldsymbol{x}$ weakly dominates $\boldsymbol{x}'$ *if, $\forall 1 \leq i \leq m, f_i(\boldsymbol{x}) \geq f_i(\boldsymbol{x}')$, denoted as $\boldsymbol{x} \succeq \boldsymbol{x}'$;*
2. $\boldsymbol{x}$ dominates $\boldsymbol{x}'$ *if, $\boldsymbol{x} \succeq \boldsymbol{x}'$ and $f_i(\boldsymbol{x}) > f_i(\boldsymbol{x}')$ for some i, denoted as $\boldsymbol{x} \succ \boldsymbol{x}'$.*

Evolutionary algorithms (EAs) have become a popular tool for multi-objective optimization, due to their population-based nature. In previous theoretical studies, GSEMO is the most widely used multi-objective EA (MOEA) [8,14,21]. As described in Algorithm 1, it first randomly selects an initial solution, then repeats the three steps (selection, mutation, acceptance) to improve the quality of the population. In selection, a solution is uniformly selected from the current population; in mutation, a new solution is generated by flipping each bit of the selected solution with probability $\frac{1}{n}$; in acceptance, the new solution is compared with the solutions in the population, and then only non-dominated solutions are kept. Although simple, GSEMO explains the common structure of various MOEAs, and hence will be used in this paper as well.

Selection hyper-heuristics manage a set of low-level heuristics, and select an appropriate one to be applied at each decision point. Despite their practical successes, the theoretical analysis is still in its infancy, particularly for multi-objective optimization. In this paper, we take the three components of GSEMO, i.e., selection, mutation and acceptance, as the low-level heuristic, respectively, and compare the performance of GSEMO with a mixed strategy and a pure strategy. For each component of GSEMO, we will use two concrete low-level heuristics. A typical mixed strategy employed in our analysis (denoted by GSEMO$_p$) is to use the first low-level heuristic with probability $p \in [0, 1]$ in each iteration of GSEMO, and use the second one otherwise. Note that a mixed strategy corresponds to using heuristic selection, while a pure strategy only uses one specific low-level heuristic and thus implies that heuristic selection is not employed.

The performance of the comparison algorithms is measured by their running time complexity. Note that running time analysis has been a leading theoretical aspect for randomized search heuristics [2,20]. The running time of a MOEA is usually counted by the number of fitness evaluations (the most costly computational process) until finding the Pareto front [8,14,21].

Algorithm 1 GSEMO

Given the solution space $\mathcal{X} = \{0,1\}^n$ and the objective function vector $\boldsymbol{f}$, GSEMO consists of the following steps:

```
1: Choose $\boldsymbol{x} \in \mathcal{X}$ uniformly at random
2: $P \leftarrow \{\boldsymbol{x}\}$
3: repeat
4:     [Selection] Choose $\boldsymbol{x}$ from $P$ uniformly at random
5:     [Mutation] Create $\boldsymbol{x}'$ by flipping each bit of $\boldsymbol{x}$ with probability $1/n$
6:     [Acceptance] if $\nexists \boldsymbol{z} \in P$ such that $\boldsymbol{z} \succ \boldsymbol{x}'$
7:             $P \leftarrow (P - \{\boldsymbol{z} \in P \mid \boldsymbol{x}' \succeq \boldsymbol{z}\}) \cup \{\boldsymbol{x}'\}$
8:         end if
9: until some criterion is met
```

3 Mixing Selection Mechanisms

In this section, we use two fair selection mechanisms [9,14] as low-level heuristics:

- **fair selection w.r.t. the decision space:** Each solution in the current population has a counter c_1, which records the number of its offsprings. The solution with the smallest c_1 value will be selected for reproduction in each iteration. That is, line 4 of Algorithm 1 changes to be "Choose $\boldsymbol{x} \in \{\boldsymbol{y} \in P \mid c_1(\boldsymbol{z}) \geq c_1(\boldsymbol{y}), \forall \boldsymbol{z} \in P\}$ uniformly at random".
- **fair selection w.r.t. the objective space:** Each counter (denoted as c_2) is associated with an objective vector rather than a decision vector. Line 4 thus changes to be "Choose $\boldsymbol{x} \in \{\boldsymbol{y} \in P \mid c_2(f(\boldsymbol{z})) \geq c_2(f(\boldsymbol{y})), \forall \boldsymbol{z} \in P\}$ uniformly at random".

Fairness is employed to balance the number of offsprings of all solutions in the current population, and thus to achieve a good spread over the Pareto front. GSEMO with these two mechanisms are denoted by GSEMO$_{ds}$ and GSEMO$_{os}$, respectively. For GSEMO with the mixed strategy (denoted by GSEMO$_p$), it uses the fairness w.r.t the decision space with probability $p \in [0,1]$ in each iteration; otherwise, it uses the fairness w.r.t the objective space. Note that GSEMO$_{ds}$ and GSEMO$_{os}$ are GSEMO$_p$ with $p = 1$ and $p = 0$, respectively.

We then compare their running time on the ZPLG function. As shown in Definition 3, ZPLG can be divided into three parts: ZeroMax, a plateau, and a path with little gaps. It is obtained from the PLG function in [9] by replacing the second objective value 1 in the ZeroMax part with 2. The Pareto front is $\{(n,2), (n+1,1), (\frac{9n}{8}+2, 0)\}$, and the corresponding Pareto optimal solutions are 0^n, SP_1 and 1^n, respectively.

Definition 3 (ZPLG).

$$ZPLG(\boldsymbol{x}) = \begin{cases} (|\boldsymbol{x}|_0, 2) & \boldsymbol{x} \notin SP_1 \cup SP_2 \\ (n+1, 1) & \boldsymbol{x} \in SP_1 \\ (n+2+i, 0) & \boldsymbol{x} = 1^{3n/4+2i}0^{n/4-2i} \in SP_2, \end{cases}$$

where $|\boldsymbol{x}|_0 = \sum_{j=1}^{n}(1-x_j)$ *denotes the number of 0-bits,* $SP_1 = \{1^i 0^{n-i} \mid 1 \leq i < 3n/4\}$*,* $SP_2 = \{1^{3n/4+2i}0^{n/4-2i} \mid 0 \leq i \leq n/8\}$ *and* $n = 8m, m \in \mathbb{N}$.

Theorem 1 shows that GSEMO with a pure strategy needs exponential running time with a high probability. The result of GSEMO$_{os}$ on ZPLG is directly from that on the PL function (i.e., Theorem 1) in [9], since ZPLG has the same structure as PL by treating its SP_2 part as a whole. The inefficiency is because GSEMO$_{os}$ allows the Pareto optimal solution 0^n to generate new solutions in SP_1, which stop the random walk on the plateau SP_1 and thus prevent from reaching SP_2. The result of GSEMO$_{ds}$ on ZPLG can be directly from that on the PLG function (i.e., Theorem 4) in [9], since their proof relies on SP_1 and SP_2, which are the same for ZPLG and PLG. The inefficiency is because GSEMO$_{ds}$ easily gets trapped in the random walk on SP_1, which prevents from following the path SP_2 to find the Pareto optimal solution 1^n. We then prove in Theorem 2 that by using the mixed strategy, GSEMO$_p$ can solve ZPLG in polynomial running time. The idea is that first employing GSEMO$_{ds}$ allows the random walk on SP_1 to reach SP_2, and then employing GSEMO$_{os}$ allows following the path SP_2 to find 1^n. Thus, we can see that the advantage of using heuristic selection is that the strengths of one heuristic can compensate for the weaknesses of another.

Theorem 1. *On ZPLG, the running time of GSEMO$_{ds}$ is $2^{\Omega(n^{1/2})}$ with probability $1 - 2^{-\Omega(n^{1/2})}$, and that of GSEMO$_{os}$ is $2^{\Omega(n^{1/4})}$ with probability $1 - e^{-\Omega(n^{1/3})}$.*

Theorem 2. *The expected running time of GSEMO$_p$ with $p = 1 - \frac{1}{n^3}$ on ZPLG is $O(n^6)$.*

Proof. We divide the optimization process into two phases: (1) starts after initialization and finishes until the population P contains 0^n, a solution from SP_1 and a solution from SP_2; (2) starts after phase (1) and finishes until P contains 0^n, a solution from SP_1 and 1^n, i.e., the Pareto front is found.

For the first phase, we can follow the analysis of GSEMO$_{ds}$ on PL (i.e., Theorem 2) in [9]. In their proof, the only part relying on the fair selection w.r.t. the decision space is to allow a consecutive random walk of length δn^3 (δ is a constant) on the plateau SP_1, under the condition that the c_1 value of the maintained solution from SP_1 is always smaller than that of the Pareto optimal solution 0^n. Note that the fair selection w.r.t. the decision space is used with probability $1 - 1/n^3$ in each iteration of GSEMO$_p$. Such a random walk happens with probability $(1 - \frac{1}{n^3})^{\delta n^3} \geq (2e)^{-\delta} \in \Omega(1)$. Thus, the asymptotic running time is not affected, and the expected running time of this phase is the same as that of GSEMO$_{ds}$ on PL, i.e., $O(n^3 \log n)$.

For the second phase, the population P always contains three solutions, 0^n, a solution from SP_1 and a solution from SP_2. The probability that a better solution from SP_2 is found under the condition that a solution from SP_2 has been selected for mutation is at least $\frac{1}{n^2}(1 - \frac{1}{n})^{n-2} \geq \frac{1}{en^2}$, since it is sufficient to flip the leftmost two 0-bits. It is easy to see that at most $\frac{n}{8}$ such improvements are sufficient to find the Pareto optimal solution 1^n. The worst case is reached when the first found solution from SP_2 is $1^{3n/4}0^{n/4}$. We consider that the fair selection w.r.t. the objective space is used, which happens with probability $\frac{1}{n^3}$ in each iteration of GSEMO$_p$. Because the c_2 values of $(n, 2)$ and $(n + 1, 1)$ (i.e., the objective vectors of 0^n and the solution from SP_1) are never decreased, the solution from SP_2 is selected for reproduction at least once in three consecutive iterations. Thus, the expected running time of this phase is at most $n^3 \cdot 3 \cdot \frac{n}{8} \cdot en^2 \in O(n^6)$. □

4 Mixing Mutation Operators

In this section, we use two mutation operators [15] as low-level heuristics:

- **one-bit mutation:** Line 5 of Algorithm 1 changes to be "Create $\boldsymbol{x}'$ by flipping one randomly chosen bit of $\boldsymbol{x}$". Note that one specific bit is chosen with probability $\frac{1}{n}$.
- **two-bit mutation:** Line 5 of Algorithm 1 changes to be "Create $\boldsymbol{x}'$ by flipping two different and randomly chosen bits of $\boldsymbol{x}$". Note that two specific bits are chosen with probability $1/\binom{n}{2} = \frac{2}{n(n-1)}$.

GSEMO with these two operators are denoted by GSEMO$_{1b}$ and GSEMO$_{2b}$, respectively. GSEMO with the mixed strategy (denoted by GSEMO$_p$) uses one-bit mutation with probability $p \in [0, 1]$ in each iteration; otherwise, it uses two-bit mutation.

We then compare their running time on the SPG function. As shown in Definition 4, SPG has a short path SP with increasing fitness except the solutions 1^i0^{n-i} with i mod $3 = 1$. The construction of SPG is inspired from the GapPath function in [15]. The Pareto front is $\{(n, 1), (n^2, 0)\}$, and the corresponding Pareto optimal solutions are 0^n and 1^n, respectively.

Definition 4 (SPG).

$$SPG(\boldsymbol{x}) = \begin{cases} (|\boldsymbol{x}|_0, 1) & \boldsymbol{x} \notin SP \\ (-1, 0) & \boldsymbol{x} = 1^i0^{n-i} \in SP,\ i \textit{ mod } 3 = 1 \\ (in, 0) & \boldsymbol{x} = 1^i0^{n-i} \in SP,\ i \textit{ mod } 3 = 0 \textit{ or } 2, \end{cases}$$

where $SP = \{1^i0^{n-i} \mid 1 \leq i \leq n\}$ *and* $n = 3m, m \in \mathbb{N}$.

The following two theorems show that the expected running time of GSEMO with a pure strategy is infinite while that of GSEMO with the mixed strategy is polynomial. The proof idea is straightforward. In every three adjacent solutions on the path SP, there is a bad one 1^i0^{n-i} with i mod $3 = 1$. Using one-bit and two-bit mutation alternatively can jump over those bad solutions on SP and finally reach the Pareto optimal solution 1^n, while using only one-bit or two-bit mutation obviously will get stuck in some solution 1^i0^{n-i} with i mod $3 = 0$ or 2.

Theorem 3. *The expected running time of GSEMO$_{1b}$ and GSEMO$_{2b}$ on SPG is infinite.*

Proof. We consider that the initial solution is the Pareto optimal solution 0^n, which has the objective vector $(n, 1)$. This happens with probability $\frac{1}{2^n}$ due to the uniform sampling. For GSEMO$_{1b}$, one-bit mutation on 0^n can only generate solutions with the objective vector $(n-1, 1)$ or $(-1, 0)$, which are dominated by 0^n. Thus, the population P will always contain only 0^n. For GSEMO$_{2b}$, two-bit mutation on 0^n generates solutions with the objective vector $(n-2, 1)$ or $(2n, 0)$. Thus, P contains 0^n and 1^20^{n-2} after a while. Since two-bit mutation on 1^20^{n-2} cannot generate better solutions on SP, P will always keep in this state. Thus, starting from 0^n, either GSEMO$_{1b}$ or GSEMO$_{2b}$ cannot find the Pareto front, which implies that the expected running time is infinite. $\square$

Theorem 4. *The expected running time of GSEMO$_p$ with* $p \in [0, 1]$ *being a constant on SPG is* $O(n^3)$.

Proof. The population P contains at most two solutions, because the second objective of SPG has only two values 0 and 1. We first analyze the expected running time until the Pareto optimal solution 0^n is found. Let $\boldsymbol{x}$ denote the solution with the second objective value 1 in P. Such a solution will exist in P after at most n expected running time. This is because a solution from SP can generate an offspring solution not from SP by flipping the first 1-bit, which happens with probability at least $\frac{1}{n}$ by either one-bit or two-bit mutation. Assume that the number of 0-bits of $\boldsymbol{x}$ is j $(j \geq 1)$. It is easy to see that j cannot decrease, and it can increase by flipping one 1-bit (but not the last) using one-bit mutation. Because the probability of selecting $\boldsymbol{x}$ for mutation is at least $\frac{1}{2}$ and one-bit mutation is used with probability p, the probability of increasing j by 1 in one iteration is at least $\frac{1}{2} \cdot p \cdot \frac{n-j-1}{n}$ for $j \leq n-2$ and $\frac{1}{2} \cdot p \cdot \frac{1}{n}$ for $j = n-1$. Thus, the expected running time to find 0^n is at most $\sum_{j=1}^{n-2} \frac{2n}{p(n-j-1)} + \frac{2n}{p} \in O(n \log n)$.

When finding 0^n, we pessimistically assume that the solution from SP has not been found. Starting from 0^n, the solution $1^2 0^{n-2}$ can be found by flipping the first two 0-bits using two-bit mutation. This happens with probability $(1-p) \cdot \frac{2}{n(n-1)}$, and thus the expected running time is $\frac{n(n-1)}{2(1-p)} \in O(n^2)$. Once a solution from SP with i mod $3 \neq 1$ has been found, using one-bit and two-bit mutation alternatively can follow the path SP to find the Pareto optimal solution 1^n. If i mod $3 = 2$, flipping its first 0-bit by one-bit mutation can generate a better solution. This happens with probability $\frac{1}{2} \cdot p \cdot \frac{1}{n}$. If i mod $3 = 0$, flipping its first two 0-bits by two-bit mutation can generate a better solution. This happens with probability $\frac{1}{2} \cdot (1-p) \cdot \frac{2}{n(n-1)}$. Since $\frac{n}{3}$ such two improvements are sufficient to find 1^n, the expected running time is at most $\frac{n}{3} \cdot (\frac{2n}{p} + \frac{n(n-1)}{(1-p)}) \in O(n^3)$. Thus, the theorem holds. $\square$

5 Mixing Acceptance Strategies

In this section, we use two acceptance strategies as low-level heuristics:

– **elitist acceptance:** As lines 6-8 of Algorithm 1, only non-dominated solutions are kept in the population, and the existing solution in P with the same objective vector as the newly generated solution will be replaced.
– **strict elitist acceptance:** It is the same as elitist acceptance, except that the old solution with the same objective vector as the newly generated solution will not be replaced. That is, line 6 of Algorithm 1 changes to be "if $\nexists \boldsymbol{z} \in P$ such that $\boldsymbol{z} \succeq \boldsymbol{x}'$".

The difference between these two strategies is to accept or reject the solution with the same fitness. This has been theoretically shown to have a significant effect on the performance of EAs in single-objective optimization [13]. Note that GSEMO with elitist acceptance is just GSEMO. GSEMO with the strict strategy is denoted by GSEMO$_{strict}$. In each iteration of GSEMO with the mixed strategy (denoted by GSEMO$_{mixed}$), elitist acceptance is used if the newly generated solution $\boldsymbol{x}'$ and the parent solution $\boldsymbol{x}$ have the same objective vector; otherwise, strict elitist acceptance is used. Note that the mixed strategy employed here is different from that of GSEMO$_p$.

We then compare their running time on the PL function. As shown in Definition 5, PL has a short path $SP - \{1^n\}$ with constant fitness. The Pareto front is $\{(n, 1), (n+2, 0)\}$, and the corresponding Pareto optimal solutions are 0^n and 1^n, respectively.

Definition 5 (PL). *[8]*

$$PL(\boldsymbol{x}) = \begin{cases} (|\boldsymbol{x}|_0, 1) & \boldsymbol{x} \notin SP = \{1^i 0^{n-i} \mid 1 \leq i \leq n\} \\ (n+1, 0) & \boldsymbol{x} \in \{1^i 0^{n-i} \mid 1 \leq i < n\} \\ (n+2, 0) & \boldsymbol{x} = 1^n. \end{cases}$$

Theorem 5 shows that GSEMO with a pure strategy on PL needs exponential running time. The result of GSEMO was proved in [8], and its inefficiency is because a solution not from SP can generate a new solution from SP, which stops the ongoing random walk on SP. The inefficiency of GSEMO$_{strict}$ is because the first found solution from SP is far from the Pareto optimal solution 1^n, and strict elitist acceptance does not allow the random walk on SP. We then prove in Theorem 6 that GSEMO with the mixed strategy can solve PL in polynomial running time. It works by allowing accepting the solution with the same fitness only in the random walk procedure.

Theorem 5. *On PL, the running time of GSEMO is* $2^{\Omega(n^{1/24})}$ *with probability* $1 - e^{-\Omega(n^{1/24})}$ *[8], and that of GSEMO$_{strict}$ is* $n^{\Omega(\frac{n}{5})}$ *with probability* $1 - 2^{-\Omega(n)}$.

Proof. The initial solution is not in SP with probability $1 - \frac{n}{2^n}$ due to uniform selection, and it has at most $\frac{2n}{3}$ 1-bits with probability $1 - e^{-\Omega(n)}$ by Chernoff bounds. The population P contains at most two solutions, since the second objective of PL has only two different values. Note that the number of 1-bits of the solution not from SP will never increase, since the first objective is to maximize the number of 0-bits. Because the probability of flipping at least $\frac{n}{12}$ bits simultaneously in one step is less than $n^{-\frac{n}{12}}$, the first found solution from SP has at most $\frac{3n}{4}$ 1-bits with probability at least $1-n^{-\frac{n}{12}}$. Once a solution from $SP-\{1^n\}$ has been found, it will never change because $SP - \{1^n\}$ is a plateau and GSEMO$_{strict}$ will not replace the solution with the same fitness. Thus, P will always contain two solutions, a solution $\boldsymbol{x}$ from $SP - \{1^n\}$ with $|\boldsymbol{x}|_1 \leq \frac{3n}{4}$ and a solution $\boldsymbol{y}$ not from SP with $|\boldsymbol{y}|_1 \leq \frac{2n}{3}$. The probabilities of mutating $\boldsymbol{x}$ and $\boldsymbol{y}$ to 1^n in one step are at most $n^{-\frac{n}{4}}$ and $n^{-\frac{n}{3}}$, respectively. Thus, after $n^{\frac{n}{5}}$ steps, the Pareto optimal solution 1^n is generated with probability at most $n^{\frac{n}{5}} \cdot n^{-\frac{n}{4}} = n^{-\frac{n}{20}}$ by the union bound. By combining all the above probabilities, we get that the running time is $n^{\Omega(\frac{n}{5})}$ with probability $1 - 2^{-\Omega(n)}$. □

Theorem 6. *The expected running time of GSEMO$_{mixed}$ on PL is* $O(n^3)$.

Proof. Since the function PL outside SP has the same structure as OneMax, the expected steps to find 0^n is $O(n \log n)$ by using the analysis result of the (1+1)-EA on OneMax [7]. Then, it needs $O(n)$ expected steps to find a solution from SP, as it suffices to flip the leftmost 0-bit of 0^n. For GSEMO$_{mixed}$, if an offspring solution from SP is generated by mutation on the solution not from SP, it will not replace the solution from SP in the current population; but if it is generated by mutation on the current solution from SP, the replacement will be implemented. Thus, the algorithm will perform the random walk on the plateau SP and the solution not from SP will not influence it. Note that the solution from SP is selected for mutation with probability $\frac{1}{2}$. Using the analysis result of the (1+1)-EA on SPC (i.e., Theorem 7) in [13], we get that the random walk needs $O(n^3)$ expected running time to find 1^n. □

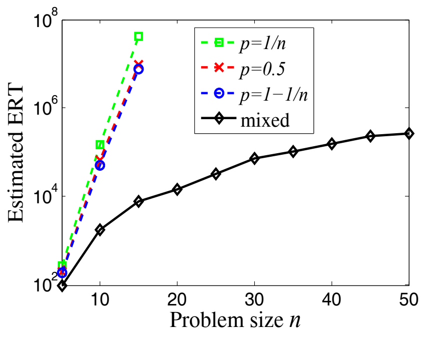

Figure 1. Estimated ERT of GSEMO$_{mixed}$ and GSEMO$_p$ for solving the PL problem, where a base 10 logarithmic scale is used for the y-axis.

Note that the mixed strategy employed by GSEMO$_{mixed}$ here is different from that by GSEMO$_p$ for mixing selection mechanisms or mutation operators. GSEMO$_p$ uses the first low-level heuristic with probability $p \in [0, 1]$ in each iteration and uses the second one otherwise. To investigate the influence of different mixed strategies, we conduct experiments to compare GSEMO$_{mixed}$ with GSEMO$_p$ for mixing elitist and strict elitist acceptance. The parameter p is set as $\frac{1}{n}$, 0.5 and $1-\frac{1}{n}$, respectively. For each comparison algorithm on each problem size $n \in \{5, 10, \ldots, 50\}$, we run the algorithm 100 times independently, where each run stops when the Pareto front of the PL problem is found. The average number of fitness evaluations is used as the estimation of the expected running time (ERT). The result is plotted in Figure 1. Note that the ERT of GSEMO$_p$ for $n \geq 20$ is too large to estimate. We can observe that GSEMO$_{mixed}$ is much more efficient than GSEMO$_p$. The curves of GSEMO$_{mixed}$ and GSEMO$_p$ grow in a closely logarithmic and linear trend, respectively, which implies that their ERT is approximately polynomial and exponential, respectively. Thus, these empirical results suggest that choosing a proper threshold selection method is important.

6 Conclusion

This paper presents a theoretical study on the effectiveness of selection hyper-heuristics for multi-objective optimization. Rigorous running time analysis showed that applying selection hyper-heuristics to any of the three major components of a MOEA, i.e., selection, mutation and acceptance, can exponentially speed up the optimization. From the analysis, we find that selection hyper-heuristics work by allowing the strengths of one heuristic to compensate for the weaknesses of another. Our result provides theoretical support for multi-objective selection hyper-heuristics. The empirical comparison on different mixed strategies also implies the importance of choosing a proper heuristic selection method.

References

1. Alanazi, F., Lehre, P.K.: Runtime analysis of selection hyper-heuristics with classical learning mechanisms. In: Proceedings of CEC'14. pp. 2515–2523. Beijing, China (2014)

2. Auger, A., Doerr, B.: Theory of Randomized Search Heuristics: Foundations and Recent Developments. World Scientific, Singapore (2011)
3. Burke, E.K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Qu, R.: Hyper-heuristics: A survey of the state of the art. Journal of the Operational Research Society 64(12), 1695–1724 (2013)
4. Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Woodward, J.R.: A classification of hyper-heuristic approaches. In: Handbook of Metaheuristics, pp. 449–468. Springer (2010)
5. Burke, E.K., Silva, J.D.L., Soubeiga, E.: Multi-objective hyper-heuristic approaches for space allocation and timetabling. In: Metaheuristics: Progress as Real Problem Solvers, pp. 129–158. Springer (2005)
6. Corus, D., He, J., Jansen, T., Oliveto, P.S., Sudholt, D., Zarges, C.: On easiest functions for somatic contiguous hypermutations and standard bit mutations. In: Proceedings of GECCO'15. pp. 1399–1406. Madrid, Spain (2015)
7. Droste, S., Jansen, T., Wegener, I.: On the analysis of the (1+1) evolutionary algorithm. Theoretical Computer Science 276(1-2), 51–81 (2002)
8. Friedrich, T., Hebbinghaus, N., Neumann, F.: Plateaus can be harder in multi-objective optimization. Theoretical Computer Science 411(6), 854–864 (2010)
9. Friedrich, T., Horoba, C., Neumann, F.: Illustration of fairness in evolutionary multi-objective optimization. Theoretical Computer Science 412(17), 1546–1556 (2011)
10. Guizzo, G., Fritsche, G.M., Vergilio, S.R., Pozo, A.T.R.: A hyper-heuristic for the multi-objective integration and test order problem. In: Proceedings of GECCO'15. pp. 1343–1350. Madrid, Spain (2015)
11. He, J., He, F., Dong, H.: Pure strategy or mixed strategy? - An initial comparison of their asymptotic convergence rate and asymptotic hitting time. In: Proceedings of EvoCOP'12. pp. 218–229. Malaga, Spain (2012)
12. He, J., Hou, W., Dong, H., He, F.: Mixed strategy may outperform pure strategy: An initial study. In: Proceedings of CEC'13. pp. 562–569. Cancun, Mexico (2013)
13. Jansen, T., Wegener, I.: Evolutionary algorithms-how to cope with plateaus of constant fitness and when to reject strings of the same fitness. IEEE Transactions on Evolutionary Computation 5(6), 589–599 (2001)
14. Laumanns, M., Thiele, L., Zitzler, E.: Running time analysis of multiobjective evolutionary algorithms on pseudo-Boolean functions. IEEE Transactions on Evolutionary Computation 8(2), 170–182 (2004)
15. Lehre, P.K., Özcan, E.: A runtime analysis of simple hyper-heuristics: To mix or not to mix operators. In: Proceedings of FOGA'13. pp. 97–104. Adelaide, Australia (2013)
16. Maashi, M., Özcan, E., Kendall, G.: A multi-objective hyper-heuristic based on choice function. Expert Systems with Applications 41(9), 4475–4493 (2014)
17. McClymont, K., Keedwell, E.C.: Markov chain hyper-heuristic (MCHH): An online selective hyper-heuristic for multi-objective continuous problems. In: Proceedings of GECCO'11. pp. 2003–2010. Dublin, Ireland (2011)
18. Miranda, G., De Armas, J., Segura, C., León, C.: Hyperheuristic codification for the multi-objective 2d guillotine strip packing problem. In: Proceedings of CEC'10. pp. 1–8. Barcelona, Spain (2010)
19. Mitavskiy, B., He, J.: A polynomial time approximation scheme for a single machine scheduling problem using a hybrid evolutionary algorithm. In: Proceedings of CEC'12. pp. 1–8. Brisbane, Australia (2012)
20. Neumann, F., Witt, C.: Bioinspired Computation in Combinatorial Optimization: Algorithms and Their Computational Complexity. Springer-Verlag, Berlin, Germany (2010)
21. Qian, C., Yu, Y., Zhou, Z.H.: An analysis on recombination in multi-objective evolutionary optimization. Artificial Intelligence 204, 99–119 (2013)