Evolutionary Multi-objective Optimization Made Faster by Sequential Decomposition

Jing-Cheng Shi
National Key Laboratory for
Novel Software Technology
Nanjing University
Nanjing 210023, China
Email: shijc@lamda.nju.edu.cn

Chao Qian
School of Computer Science and Technology
University of Science and Technology of China
Hefei 230027, China
Email: chaoqian@ustc.edu.cn

Yang Yu
National Key Laboratory for
Novel Software Technology
Nanjing University
Nanjing 210023, China
Email: yuy@nju.edu.cn

***Abstract*—Multi-objective evolutionary algorithms (MOEAs) can be mainly divided into set approximation methods and decomposition methods. The former approximates the Pareto front by the whole population directly, while the latter solves decomposed subproblems. The theoretical understanding of these methods is, however, quite insufficient. In this paper, we try to gain more understanding by investigating a combination of set approximation MOEAs with a sequential decomposition mechanism. Our theoretical analysis shows that, the combination achieves a better running time than the corresponding set approximation MOEAs by a factor n (the problem size) on synthetic problems as well as the minimum spanning tree problem, which hints that the two types of MOEAs might be mutually complemental.**

I. Introduction

In real-world optimization tasks, we often need to consider two or more objective functions. The optimized objectives are usually conflicting, and thus multi-objective optimization is to find a set of Pareto optimal solutions, which represent different trade-offs between objectives. Evolutionary algorithms (EAs) [1] are a kind of randomized heuristic optimization algorithms, inspired by nature phenomena. They maintain a set of solutions (called a population), and repeatedly try to improve the population by using genetic operators (e.g., mutation and crossover). Due to their population-based nature, EAs have been widely and successfully applied for solving multi-objective optimization problems [2].

Previous multi-objective EAs (MOEAs) can be mainly divided into two categories: set approximation methods and decomposition methods. The former approximates the Pareto front by the whole population directly, e.g., SPEA-II [3] and NSGA-II [4]. The latter first decomposes a multi-objective optimization problem into a number of single-objective (or multi-objective) optimization subproblems, and then employs a population-based method to optimize these subproblems simultaneously. The decomposition strategy was introduced into MOEAs early, e.g., [5], [6]. The most popular MOEA with a decomposition strategy is the MOEA/D algorithm [7], which uses traditional aggregation methods to construct a series of single-objective subproblems. Many variants of MOEA/D have been developed, e.g., the decomposition based memetic algorithm [8] and the MOEA/D-M2M algorithm [9]. A literature review on MOEA/D can be seen in [10].

The theoretical analysis of MOEAs is difficult due to their complexity and randomness. For the running time analysis (one essential theoretical aspect [11], [12]), only a few pieces of studies have been reported. Laumanns et al. [13], [14] first analyzed the running time of SEMO and GSEMO (two simple MOEAs) on two artificial bi-objective pseudo-Boolean problems LOTZ and COCZ. Later, the running time bounds were derived for solving some combinatorial optimization problems, e.g., minimum spanning tree [15], minimum set cover [16], minimum cuts [17] and minimum cost coverage [18]. Qian et al. [19] proved the effectiveness of selection hyper-heuristics for GSEMO solving some artificial problems. That is, using selection hyper-heuristics can reduce the running time from exponential to polynomial. Note that SEMO and GSEMO do not involve crossover operators. The helpfulness of crossover was then proved for MOEAs solving the multi-criteria all-pairs-shortest-path problem [20] as well as the multi-objective minimum spanning tree problem [21]. Recently, the running time of SMO-GP (a simple multi-objective genetic programming algorithm) has been analyzed on the Order, Majority and Sorting problems [22], [23], [24] as well as the minimum spanning tree problem [25]. The MOEAs investigated in the above-mentioned studies do not use a decomposition strategy. To the best of our knowledge, the only theoretical work on decomposition based MOEAs is that Li et al. [26] analyzed the running time of MOEA/D on some simple artificial bi-objective functions.

In this paper, we try to gain more understanding of MOEAs by investigating a combination of set approximation MOEAs with a sequential decomposition mechanism. The original multi-objective problem is first decomposed into a set of ordered multi-objective subproblems according to the range of values on some selected objective, and then an existing set approximation MOEA is employed to solve these subproblems sequentially. We show that using sequential decomposition can make MOEAs faster by rigorous running time analysis. Particularly, the running time of an MOEA with and without sequential decomposition is compared for solving synthetic problems as well as the minimum spanning tree problem.

Problem	SEMO/GSEMO	SEMO/GSEMO-SD
LOTZ	$\Theta(n^3)$ [14]/$O(n^3)$ [13]	$O(n^2)$
Minimum Spanning Tree	$O(mn(n+\log w_{\max}))$ [15]	$O(m(n+\log w_{\max}))$

Problem	SMO-GP	SMO-GP-SD
Sorting	$O(nT_{init}+n^4)$ [24]	$O(T_{init}+n^3)$

Table I
COMPARISON BETWEEN THE EXPECTED RUNNING TIME OF AN MOEA WITH AND WITHOUT SEQUENTIAL DECOMPOSITION, WHERE MOEA-SD DENOTES THE MOEA WITH SEQUENTIAL DECOMPOSITION.

As shown in Table I, sequential decomposition can bring a speedup of n, where n is the problem size. Our theoretical results imply that the two types of MOEAs might be mutually complemental.

The rest of this paper is organized as follows. Section II introduces some preliminaries. Section III introduces the proposed decomposition approach, the effectiveness of which is then theoretically analyzed in Section IV. Section V concludes the paper.

II. EVOLUTIONARY MULTI-OBJECTIVE OPTIMIZATION

Multi-objective optimization requires to simultaneously optimize two or more objective functions, as shown in Definition 1. Note that maximization is considered since minimizing f is equivalent to maximizing $-f$. 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 $\boldsymbol{x}$ 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 objective vector in the Pareto front.

Definition 1 (Multi-Objective Optimization). *Given a feasible solution space $\mathcal{X}$ and m objective functions $f_1, \ldots, f_m$ with $f_i : \mathcal{X} \to \mathbb{R}$, multi-objective optimization is to solve the following problem:*

$$\max_{\boldsymbol{x}\in\mathcal{X}} \quad \big(f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), ..., f_m(\boldsymbol{x})\big). \tag{1}$$

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, $\forall 1 \le i \le m, f_i(\boldsymbol{x}) \ge f_i(\boldsymbol{x}')$, 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. In previous theoretical analyses of multi-objective EAs (MOEAs), a general framework, as described in Algorithm 1, is widely used [14], [22], [21]. It first randomly selects an initial solution, and then tries to repeatedly improve the quality of the solutions in the population. In each iteration, a solution uniformly selected from the current population is first used to generate a new solution by applying mutation; then the newly generated solution is compared with the solutions in the population, and only non-dominated solutions are kept. Although simple, Algorithm 1 explains the common structure of various MOEAs, and hence will also be used in our theoretical analysis.

Algorithm 1 A Simple Framework of MOEA

Given a solution space $\mathcal{X}$ and an objective vector $\boldsymbol{f}$, the procedure:

```
1: Choose $\boldsymbol{x} \in \mathcal{X}$ uniformly at random
2: $P \leftarrow \boldsymbol{x}$
3: repeat
4:    Choose $\boldsymbol{x}$ from $P$ uniformly at random
5:    $\boldsymbol{x}' :=$ mutation($\boldsymbol{x}$)
6:    if $\nexists \boldsymbol{z} \in P$ such that $\boldsymbol{z} \succ \boldsymbol{x}'$
7:        $P \leftarrow (P \setminus \{\boldsymbol{z} \in P \mid \boldsymbol{x}' \succeq \boldsymbol{z}\}) \cup \{\boldsymbol{x}'\}$
8:    end if
9: until some criterion is met
```

III. THE PROPOSED SEQUENTIAL DECOMPOSITION APPROACH

In this section, we propose a sequential decomposition approach for evolutionary multi-objective optimization, called MOEA-SD. The main idea is to decompose the original problem into a set of ordered multi-objective subproblems according to the range of values on some objective, and then employ an existing set approximation MOEA to solve these subproblems sequentially.

The MOEA-SD approach is described in Algorithm 2. It first selects one objective $f \in \{f_1, f_2, \ldots, f_m\}$, and then divides the range of values on f into N intervals $[l_1, u_1], \ldots, [l_N, u_N]$, where $l_{i-1} < l_i \le u_{i-1} < u_i$, l_1 and u_N are the lower and upper bounds of f, respectively. The original problem Eq. (1) is then decomposed into N subproblems, where the i-th subproblem is

$$\begin{aligned} \max_{\boldsymbol{x}\in\mathcal{X}} \quad & \big(f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), ..., f_m(\boldsymbol{x})\big) \\ s.t. \quad & f(\boldsymbol{x}) \in [l_i, u_i]. \end{aligned} \tag{2}$$

That is, the i-th subproblem requires the value on the selected objective f to be in the interval $[l_i, u_i]$. After the decomposition, an existing MOEA is employed to solve these subproblems sequentially according to the search direction (forward or backward). The solutions found for each subproblem are finally combined, and only non-dominated solutions are kept as the output for the original problem.

For the forward search, the subproblems are solved from $i = 1$ to N sequentially. If $i = 1$, the solutions satisfying $f(\boldsymbol{x}) \in [l_1, u_1]$ are randomly generated as the initial population. For the i-th subproblem with $i \geq 2$, motivated by the assumption that adjacent Pareto optimal solutions on one objective might be easily reached from each other, the solutions with $f(\boldsymbol{x}) \geq l_i$ found for the $(i-1)$-th subproblem are used as the initial population. Note that the requirement that the adjacent intervals are overlapping (i.e., $l_i \leq u_{i-1}$) makes it possible to find good solutions with $f(\boldsymbol{x}) \geq l_i$ in the process of solving the $(i-1)$-th subproblem.

For the backward search, the subproblems are solved from $i = N$ to 1 sequentially. If $i = N$, the solutions satisfying $f(\boldsymbol{x}) \in [l_N, u_N]$ are randomly generated as the initial population. For the i-th subproblem with $i \leq N-1$, the solutions with $f(\boldsymbol{x}) \leq u_i$ found for the $(i+1)$-th subproblem are used as the initial population.

Note that MOEA-SD can be viewed as a combination of set approximation MOEAs with a sequential decomposition mechanism. It solves the decomposed multi-objective subproblems sequentially by using an existing set approximation MOEA. MOEA-SD is different from previous decomposition based MOEAs, which solve the single (or multi)-objective subproblems simultaneously by designing a population-based method.

Algorithm 2 MOEA-SD

Input: a multi-objective optimization problem $\boldsymbol{f} = (f_1, f_2, \ldots, f_m)$ and an existing MOEA $\mathcal{A}$
Parameter: one selected objective $f \in \{f_1, f_2, \ldots, f_m\}$, a division of the range of f: $\{[l_i, u_i] \mid 1 \leq i \leq N, l_{i-1} < l_i \leq u_{i-1} < u_i\}$, a search direction: forward or backward
Output: a set of non-dominated solutions
Process:

```
1: if forward search
2:   for $i = 1, 2, ..., N$
3:     Apply $\mathcal{A}$ to solve Eq. (2)
4:     if $i = 1$
5:       The solutions with $f(\boldsymbol{x}) \in [l_1, u_1]$ are randomly generated as the initial population
6:     else
7:       The solutions with $f(\boldsymbol{x}) \geq l_i$ in $R_{i-1}$ are used as the initial population
8:     end if
9:     Let $R_i$ denote the set of solutions output in this phase
10:  end for
11: else
12:  for $i = N, N-1, ..., 1$
13:    Apply $\mathcal{A}$ to solve Eq. (2)
14:    if $i = N$
15:      The solutions with $f(\boldsymbol{x}) \in [l_N, u_N]$ are randomly generated as the initial population
16:    else
17:      The solutions with $f(\boldsymbol{x}) \leq u_i$ in $R_{i+1}$ are used as the initial population
18:    end if
19:    Let $R_i$ denote the set of solutions output in this phase
20:  end for
21: end if
22: return the non-dominated solutions in $R_1 \cup R_2 \cup \ldots \cup R_N$
```

IV. Theoretical Analysis

In this section, we compare the running time of an MOEA with and without sequential decomposition for solving a problem. Note that running time analysis has been a leading theoretical aspect for randomized search heuristics [11], [12]. With the goal of finding the Pareto front, the running time of an MOEA is counted by the number of fitness evaluations (the most costly computational process) until finding at least one corresponding solution for each objective vector in the Pareto front [14], [21].

A. SEMO/GSEMO on LOTZ

We first consider that SEMO and GSEMO are used for solving the bi-objective pseudo-Boolean problem LOTZ (Leading Ones Trailing Zeros). SEMO and GSEMO are two simple MOEAs for multi-objective optimization over the Boolean solution space $\mathcal{X} = \{0,1\}^n$. They have the same structure as Algorithm 1, and the only difference is the mutation operator. SEMO uses one-bit mutation, i.e., line 5 of Algorithm 1 is "Create $\boldsymbol{x}'$ by flipping a randomly chosen bit of $\boldsymbol{x}$", while GSEMO uses bit-wise mutation, i.e., "Create $\boldsymbol{x}'$ by flipping each bit of $\boldsymbol{x}$ with probability $1/n$". These two algorithms have been widely used in previous theoretical analyses of MOEAs [14], [15], [21].

For LOTZ as presented in Definition 3, the first objective LO is to maximize the number of leading 1-bits, and the other objective TZ is to maximize the number of trailing 0-bits. The Pareto front is $\{(i, n-i) \mid 0 \leq i \leq n\}$, and the corresponding Pareto optimal solutions are $\{0^n, 10^{n-1}, \ldots, 1^{n-1}0, 1^n\}$.

Definition 3 (LOTZ [14])**.** *The pseudo-Boolean function LOTZ: $\{0,1\}^n \to \mathbb{N}^2$ is defined as follows:*

$$\mathit{LOTZ}(\boldsymbol{x}) = \left(\sum\nolimits_{i=1}^{n} \prod\nolimits_{j=1}^{i} x_j, \sum\nolimits_{i=1}^{n} \prod\nolimits_{j=i}^{n} (1 - x_j)\right),$$

where $x_j \in \{0,1\}$ is the j-th bit of $\boldsymbol{x}$.

We prove in Theorem 1 that the expected running time of SEMO/GSEMO using sequential decomposition is $O(n^2)$, which decreases by a factor n from that of SEMO/GSEMO (i.e., $\Theta(n^3)$ [14] and $O(n^3)$ [13]). The proof idea is to analyze the expected running time for solving each subproblem and then sum up them to get the total expected running time.

Theorem 1. *For SEMO/GSEMO-SD on LOTZ, if the selected objective is the first objective LO, the division satisfies that $u_{i-1} = l_i \wedge u_i - l_i = c$ with c being a constant, and the search direction is backward, then the expected running time is $O(n^2)$.*

Proof. Since the backward search is used, the subproblems will be solved from $i = N$ to 1 sequentially. Note that the possible values of the selected objective LO are $\{0, 1, \dots, n\}$. Because the population contains at most one solution for each LO value and $u_i - l_i = c$, the population size for solving each subproblem is always upper bounded by $c + 1$. For the simplicity of analysis, we assume that $n = Nc$.

For solving the N-th subproblem, it is to find the Pareto optimal solutions with the LO value from $l_N = n - c$ to $u_N = n$. We first analyze the expected running time until the Pareto optimal solution with the largest LO value (i.e., 1^n) is found. Let J denote the largest LO value of the solutions in the current population, and let $\boldsymbol{x}$ denote the corresponding solution. It is easy to see that J cannot decrease, since a solution with a smaller LO value cannot dominate a solution with a larger LO value. By selecting $\boldsymbol{x}$ for mutation and flipping only its first 0-bit, J can increase by at least 1. The probability of selecting $\boldsymbol{x}$ is at least $\frac{1}{c+1}$, since the population size is not larger than $c + 1$ and a solution is uniformly selected at random (i.e., line 4 of Algorithm 1). The probability of flipping only a specific bit is $\frac{1}{n}$ for SEMO, and $\frac{1}{n}(1 - \frac{1}{n})^{n-1} \geq \frac{1}{en}$ for GSEMO. Thus, J can increase in one step with probability at least $\frac{1}{(c+1)en}$. Note that the initial solution in this phase has the LO value in $[l_N, u_N]$. Thus, J must be at least $l_N = n - c$, which implies that increasing J by c times is sufficient to reach the solution 1^n. We then get that the expected running time for finding 1^n is at most $c(c+1)en$. After the Pareto optimal solution 1^i0^{n-i} has been found, $1^{i-1}0^{n-i+1}$ can be generated by flipping only the last 1-bit, the probability of which is at least $\frac{1}{(c+1)en}$. Note that a Pareto optimal solution will never be lost once found. Thus, by combining the running time of finding 1^n, $1^n \to 1^{n-1}0$, $\cdots$, $1^{n-c+1}0^{c-1} \to 1^{n-c}0^c$, we get that the expected running time for solving the N-th subproblem is at most $c(c + 1)en + c \cdot (c + 1)en$.

For solving the i-th subproblem with $i \leq N - 1$, it is to find the Pareto optimal solutions with the LO value from $(i-1)c$ to ic. Since the Pareto optimal solution with the LO value ic has been found in the process of solving the $(i+1)$-th subproblem and will be used as the initial solution in this phase, it only needs to find the remaining c Pareto optimal solutions. Using the above analysis of reaching $1^{i-1}0^{n-i+1}$ from 1^i0^{n-i}, we can easily derive that the expected running time of this phase is at most $c \cdot (c + 1)en$.

By combining the running time for solving each subproblem, the expected running time for finding the Pareto front is at most $c(c + 1)en + N \cdot c \cdot (c + 1)en$, i.e., $O(n^2)$. □

B. SEMO/GSEMO on Minimum Spanning Tree

We then consider that SEMO/GSEMO is used for solving the combinatorial optimization problem, minimum spanning tree (MST). The MST problem can be described as follows. Given an undirected connected graph $G = (V, E)$ on n vertices and m edges, where V and E are the vertex set and edge set, respectively, the MST problem is to find a subgraph of G with the minimum weight, which connects all the vertices. A subgraph can be represented as a Boolean solution $\boldsymbol{x} \in \{0, 1\}^m$, where $x_i = 1$ means that the i-th edge is selected. Let $w_i > 0$ denote the weight of the i-th edge and let $w_{\max} = \max\{w_i \mid 1 \leq i \leq m\}$. The MO-MST problem is presented in Definition 4, which requires minimizing the weight and the number of connected components of a subgraph simultaneously.

Definition 4 (MO-MST [15])**.** *The bi-objective minimum spanning tree problem MO-MST:* $\{0, 1\}^m \to \mathbb{N}^2$ *is defined as follows:*

$$\textit{MO-MST}(\boldsymbol{x}) = \left(\sum\nolimits_{i=1}^{m} w_i x_i, C(\boldsymbol{x})\right),$$

where $C(\boldsymbol{x})$ *is the number of connected components and* w_i *is a positive integer.*

The Pareto front of MO-MST is $\{(W_i, i) \mid 1 \leq i \leq n, W_i$ is the minimum weight among all possible subgraphs with i connected components$\}$. We prove in Theorem 2 that SEMO/GSEMO-SD can solve MO-MST in $O(m(n + \log w_{\max}))$ expected running time. Compared with that of SEMO/GSEMO (i.e., $O(mn(n + \log w_{max}))$ [15]), using sequential decomposition brings a speedup of n.

Theorem 2. *For SEMO/GSEMO-SD on MO-MST, if the selected objective is* $C(\boldsymbol{x})$*, the division satisfies that* $u_{i-1} = l_i \wedge u_i - l_i = c$ *with* c *being a constant, and the search direction is backward, then the expected running time is* $O(m(n + \log w_{\max}))$.

Proof. The subproblems will be solved from $i = N$ to 1 sequentially due to the backward search. Note that the possible values of the selected objective $C(\boldsymbol{x})$ are $\{1, \dots, n\}$. For the simplicity of analysis, we assume that $n - 1 = Nc$.

For solving the N-th subproblem, it is to find the Pareto optimal solutions with the $C(\boldsymbol{x})$ value from $n-c$ to n. We first derive the upper bound $2e(c+1)\lceil(\log 2)m(\log m+\log w_{\max}+1)\rceil \in O(m(\log n + \log w_{\max}))$ on the expected running time until finding the Pareto optimal solution with $C(\boldsymbol{x}) = n$ (i.e., the empty graph). The proof is similar to Theorem 1 in [15], which derives an upper bound on the expected running time of SEMO/GSEMO for finding the empty graph. The only difference is that the probability of selecting a specific solution for mutation is at least $\frac{1}{c+1}$ instead of $\frac{1}{n}$, because the population size is upper bounded by $c + 1$ here instead of n in [15]. We know that a subgraph with the minimum weight among all possible subgraphs with $i-1$ connected components can be found by inserting the lightest edge which will not lead to a cycle into a subgraph with the minimum weight among all possible subgraphs with i connected components. Thus, after finding a Pareto optimal solution with $C(\boldsymbol{x}) = i$, a Pareto optimal solution with $C(\boldsymbol{x}) = i - 1$ can be generated in one step with probability at least $\frac{1}{c+1} \cdot \frac{1}{em}$, because the probability of selecting a specific solution is at least $\frac{1}{c+1}$ and the probabilities of flipping only one specific bit for SEMO and GSEMO are $\frac{1}{m}$ and $\frac{1}{m}(1 - \frac{1}{m})^{m-1} \geq \frac{1}{em}$, respectively. We then get that the expected running time of this phase is at most $O(m(\log n + \log w_{\max})) + c \cdot (c + 1)em$.

For solving the i-th subproblem with $i \leq N-1$, it is to find the Pareto optimal solutions with the $C(\boldsymbol{x})$ value from $(i-1)c+1$ to $ic+1$. Since the Pareto optimal solution with $C(\boldsymbol{x}) = ic+1$ found in the previous phase will be used as the initial solution of this phase, it only needs to find the remaining c Pareto optimal solutions. Thus, the expected running time of this phase is at most $c \cdot (c+1)em$.

Therefore, the total expected running time is at most $O(m(\log n + \log w_{\max})) + N \cdot c \cdot (c+1)em$, i.e., $O(m(n + \log w_{\max}))$. $\square$

C. SMO-GP-single/multi on Sorting

In this subsection, genetic programming (GP) using variable-length representations is considered. A solution is usually represented by the tree data structure. Given a set of functions F (e.g., arithmetic operators) and a set of terminals T (e.g., variables), an internal node of the tree denotes a function in F and a leaf node denotes a terminal in T. SMO-GP is a widely used multi-objective GP in previous theoretical analyses [22], [23], [24]. It has the same structure as Algorithm 1. The employed mutation operator is described in Definition 5. It applies one of the three operators, substitution, insertion and deletion, uniformly at random and repeats this process k times independently. For SMO-GP-single, $k = 1$, and for SMO-GP-multi, k is a random number sampled from $1 + Pois(1)$, where $Pois(1)$ is the Poisson distribution with the parameter value $\lambda = 1$. Note that the arity of each function in F studied in this paper is 2.

Definition 5 (HVL-Prime Mutation)**.** *Repeat the following procedure k times independently.* Procedure*: Apply one of the following three operators uniformly at random.*

- ***[Substitution]** Replace a randomly chosen leaf node of the solution with a new node selected randomly from T.*
- ***[Insertion]** Select a node v of the solution randomly, select a node u from F randomly, and select a node w from T randomly. Replace v with u whose children are v and w, the order of which is random.*
- ***[Deletion]** Select a leaf node v of the solution randomly, the parent and the sibling of which are p and u, respectively. Replace p with u, and delete p and v.*

We consider that SMO-GP is used for solving the sorting problem, where $F = \{J\}$ and $T = \{1, 2, \ldots, n\}$. The HAM function as described in Definition 6 is one commonly used sortedness measure, which counts the number of elements at the correct position. In [24], the sorting problem has been formulated as a bi-objective optimization problem MO-HAM as presented in Definition 7, which requires maximizing HAM and minimizing the complexity of the tree simultaneously.

Definition 6 (HAM [27])**.** *The procedure:*

1) *Parse the tree $\boldsymbol{x}$ inorder, and add the leaf nodes into a list l;*
2) *Parse l from left to right, and add the leaf into a list P if it has not yet been in P;*
3) *$HAM(\boldsymbol{x}) = |\{i \in P \mid Pos(i) = i\}|$, where $Pos(i)$ is the position of i in P.*

Definition 7 (MO-HAM [24])**.** *The bi-objective sorting problem MO-HAM: $\mathcal{X} \to \mathbb{N}^2$ is defined as follows:*

$$\textit{MO-HAM}(\boldsymbol{x}) = \big(\textit{HAM}(\boldsymbol{x}), C(\boldsymbol{x})\big),$$

where $C(\boldsymbol{x})$ is the complexity of $\boldsymbol{x}$, i.e., the number of nodes of $\boldsymbol{x}$.

The Pareto front of MO-HAM is $\{(0,0)\} \cup \{(i, 2i-1) \mid 1 \leq i \leq n\}$. For $(i, 2i-1)$, the corresponding Pareto optimal solution is a tree with i leaf nodes $1, 2, \ldots, i$ from left to right, and for $(0,0)$, the corresponding solution is the empty tree. Let T_{init} denote the number of nodes of the initial solution. We prove in Theorem 3 that the expected running time of SMO-GP-single/multi-SD for solving MO-HAM is $O(T_{init} + n^3)$. Compared with the expected running time $O(nT_{init} + n^4)$ of SMO-GP-single/multi [24], using sequential decomposition has reduced the expected running time by a factor n.

Theorem 3. *For SMO-GP-single/multi-SD on MO-HAM, if the selected objective is HAM, the division satisfies that $u_{i-1} = l_i \wedge u_i - l_i = c$ with c being a constant, and the search direction is forward, then the expected running time is $O(T_{init} + n^3)$.*

Proof. The subproblems will be solved from $i = 1$ to N sequentially due to the forward search. Note that the possible values of the selected objective HAM are $\{0, 1, \ldots, n\}$. For the simplicity of analysis, we assume that $n = Nc$.

For solving the first subproblem, it is to find the Pareto optimal solutions with the HAM value from 0 to c. Let L denote the minimum complexity of the solutions in the current population, and let $\boldsymbol{x}$ denote the corresponding solution. It is easy to see that L cannot increase, and can decrease by selecting $\boldsymbol{x}$ for mutation and applying deletion once. The probability of selecting $\boldsymbol{x}$ is at least $\frac{1}{c+1}$ since the population size is not larger than $c+1$. The probability of applying deletion once in mutation is $\frac{1}{3}$ for SMO-GP-single, and $\frac{1}{e} \cdot \frac{1}{3}$ for SMO-GP-multi. Thus, the expected number of steps for decreasing L is at most $3e(c+1)$. We then get that the expected running time for finding the empty tree with the HAM value 0 is at most $3e(c+1)T_{init}$. After finding the Pareto optimal solution $\boldsymbol{x}$ with HAM$(\boldsymbol{x}) = i$, the Pareto optimal solution with the HAM value $i+1$ can be generated by inserting the element $i+1$ into $\boldsymbol{x}$ at the correct position. The probability of selecting $\boldsymbol{x}$ is at least $\frac{1}{c+1}$, and the probability of applying insertion once in mutation is $\frac{1}{3}$ for SMO-GP-single and $\frac{1}{3e}$ for SMO-GP-multi. For insertion, the probability of selecting the element $i+1$ from T is $\frac{1}{n}$ and the probability of inserting at the correct position is at least $\frac{1}{2n-1} \cdot \frac{1}{2}$, where $\frac{1}{2n-1}$ is a lower bound on the probability of selecting the proper node from the current tree for insertion and $\frac{1}{2}$ is the probability of inserting before or after the node. Thus, the probability of generating the Pareto optimal solution with the HAM value $i+1$ in one step is at least $\frac{1}{6e(c+1)n(2n-1)}$. We then get that the expected running time of this phase is at most $3e(c+1)T_{init} + c \cdot 6e(c+1)n(2n-1)$.

For solving the i-th subproblem with $i \geq 2$, it is to find the Pareto optimal solutions with the HAM value from $(i-1)c$ to ic. Since the Pareto optimal solution with the HAM value $(i-1)c$ found in the previous phase will be used as the initial solution of this phase, it is easy to see that the expected running time for finding the remaining c Pareto optimal solutions is at most $c \cdot 6e(c+1)n(2n-1)$.

By combining the running time of each phase, we get that the total expected running time is at most $3e(c+1)T_{init} + N \cdot c \cdot 6e(c+1)n(2n-1)$, i.e., $O(T_{init} + n^3)$. $\square$

V. Conclusion

This paper theoretically investigates the effectiveness of combining set approximation MOEAs with a sequential decomposition mechanism. We prove that using sequential decomposition can decrease the expected running time of a set approximation MOEA by a factor n (the problem size) on synthetic problems as well as the minimum spanning tree problem. Our results might be helpful for designing efficient MOEAs in practice. However, the effectiveness of the proposed sequential decomposition approach has been only verified with very specific MOEAs and problems. In the future, we will try to apply sequential decomposition for real MOEAs solving real-world multi-objective optimization problems.

Acknowledgment

We would like to thank the anonymous reviewers for their valuable comments. This work was supported by the NSFC (61603367), the Jiangsu Science Foundation (BK20160066), the Fundamental Research Funds for the Central Universities (WK2150110002) and the CCF-Tencent Open Research Fund.

References

[1] T. Bäck, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford, UK: Oxford University Press, 1996.

[2] C. Coello and G. Lamont, *Applications of Multi-Objective Evolutionary Algorithms*. Singapore: World Scientific, 2004.

[3] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization," in *Proceedings of the 4th Internatioal Conference on Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN'01)*, Athens, Greece, 2001, pp. 95–100.

[4] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[5] A. Jaszkiewicz, "Genetic local search for multi-objective combinatorial optimization," *European Journal of Operational Research*, vol. 137, no. 1, pp. 50–71, 2002.

[6] ——, "On the performance of multiple-objective genetic local search on the 0/1 knapsack problem-a comparative experiment," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 4, pp. 402–412, 2002.

[7] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[8] Y. Mei, K. Tang, and X. Yao, "Decomposition-based memetic algorithm for multiobjective capacitated arc routing problem," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 2, pp. 151–165, 2011.

[9] H.-L. Liu, F. Gu, and Q. Zhang, "Decomposition of a multiobjective optimization problem into a number of simple multiobjective subproblems," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 3, pp. 450–455, 2014.

[10] A. Trivedi, D. Srinivasan, K. Sanyal, and A. Ghosh, "A survey of multiobjective evolutionary algorithms based on decomposition," *IEEE Transactions on Evolutionary Computation*, 2016, DOI: 10.1109/TEVC.2016.2608507.

[11] A. Auger and B. Doerr, *Theory of Randomized Search Heuristics: Foundations and Recent Developments*. Singapore: World Scientific, 2011.

[12] F. Neumann and C. Witt, *Bioinspired Computation in Combinatorial Optimization: Algorithms and Their Computational Complexity*. Berlin, Germany: Springer-Verlag, 2010.

[13] O. Giel, "Expected runtimes of a simple multi-objective evolutionary algorithm," in *Proceedings of the 2003 IEEE Congress on Evolutionary Computation (CEC'03)*, Canberra, Australia, 2003, pp. 1918–1925.

[14] M. Laumanns, L. Thiele, and E. Zitzler, "Running time analysis of multiobjective evolutionary algorithms on pseudo-Boolean functions," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 2, pp. 170–182, 2004.

[15] F. Neumann and I. Wegener, "Minimum spanning trees made easier via multi-objective optimization," *Natural Computing*, vol. 5, no. 3, pp. 305–319, 2006.

[16] T. Friedrich, J. He, N. Hebbinghaus, F. Neumann, and C. Witt, "Approximating covering problems by randomized search heuristics using multi-objective models," *Evolutionary Computation*, vol. 18, no. 4, pp. 617–633, 2010.

[17] F. Neumann, J. Reichel, and M. Skutella, "Computing minimum cuts by randomized search heuristics," *Algorithmica*, vol. 59, no. 3, pp. 323–342, 2011.

[18] C. Qian, Y. Yu, and Z.-H. Zhou, "On constrained Boolean Pareto optimization," in *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI'15)*, Buenos Aires, Argentina, 2015, pp. 389–395.

[19] C. Qian, K. Tang, and Z.-H. Zhou, "Selection hyper-heuristics can provably be helpful in evolutionary multi-objective optimization," in *Proceedings of the 14th International Conference on Parallel Problem Solving from Nature (PPSN'16)*, Edinburgh, Scotland, 2016, pp. 835–846.

[20] F. Neumann and M. Theile, "How crossover speeds up evolutionary algorithms for the multi-criteria all-pairs-shortest-path problem," in *Proceedings of the 11th International Conference on Parallel Problem Solving from Nature (PPSN'10)*, Krakow, Poland, 2010, pp. 667–676.

[21] C. Qian, Y. Yu, and Z.-H. Zhou, "An analysis on recombination in multi-objective evolutionary optimization," *Artificial Intelligence*, vol. 204, pp. 99–119, 2013.

[22] F. Neumann, "Computational complexity analysis of multi-objective genetic programming," in *Proceedings of the 14th ACM Annual Conference on Genetic and Evolutionary Computation (GECCO'12)*, Philadelphia, PA, 2012, pp. 799–806.

[23] A. Nguyen, T. Urli, and M. Wagner, "Single- and multi-objective genetic programming: New bounds for weighted order and majority," in *Proceedings of the 12th International Workshop on Foundations of Genetic Algorithms (FOGA'13)*, Adelaide, Australia, 2013, pp. 161–172.

[24] M. Wagner and F. Neumann, "Parsimony pressure versus multi-objective optimization for variable length representations," in *Proceedings of the 12th International Conference on Parallel Problem Solving from Nature (PPSN'12)*, Taormina, Italy, 2012, pp. 133–142.

[25] C. Qian, Y. Yu, and Z.-H. Zhou, "Variable solution structure can be helpful in evolutionary optimization," *Science China: Information Sciences*, vol. 58, no. 11, pp. 1–17, 2015.

[26] Y.-L. Li, Y. Zhou, Z.-H. Zhan, and J. Zhang, "A primary theoretical study on decomposition based multiobjective evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 4, pp. 563–576, 2016.

[27] J. Scharnow, K. Tinnefeld, and I. Wegener, "The analysis of evolutionary algorithms on sorting and shortest paths problems," *Journal of Mathematical Modelling and Algorithms*, vol. 3, no. 4, pp. 349–366, 2004.