Better Running Time of the Non-dominated Sorting Genetic Algorithm II (NSGA-II) by Using Stochastic Tournament Selection*

Chao Bian and Chao Qian

State Key Laboratory for Novel Software Technology,
Nanjing University, Nanjing 210023, China
{bianc,qianc}@lamda.nju.edu.cn

Abstract. Evolutionary algorithms (EAs) have been widely used to solve multi-objective optimization problems, and have become the most popular tool. However, the theoretical foundation of multi-objective EAs (MOEAs), especially the essential theoretical aspect, i.e., running time analysis, is still largely underdeveloped. The few existing theoretical works mainly considered simple MOEAs, while the non-dominated sorting genetic algorithm II (NSGA-II), probably the most influential MOEA, has not been analyzed except for a very recent work considering a simplified variant without crossover. In this paper, we present a running time analysis of the standard NSGA-II for solving LOTZ, the commonly used bi-objective optimization problem. Specifically, we prove that the expected running time (i.e., number of fitness evaluations) is $O(n^3)$ for LOTZ, which is the same as that of the previously analyzed simple MOEAs, GSEMO and SEMO, as well as the NSGA-II without crossover. Next, we introduce a new parent selection strategy, stochastic tournament selection (i.e., k tournament selection where k is uniformly sampled at random), to replace the binary tournament selection strategy of NSGA-II, decreasing the upper bound on the required expected running time to $O(n^2)$. Experiments are also conducted, suggesting that the derived running time upper bounds are tight. We also empirically compare the performance of the NSGA-II using the two selection strategies on the widely used benchmark problem ZDT1, and the results show that stochastic tournament selection can help the NSGA-II converge faster.

1 Introduction

Multi-objective optimization, which requires optimizing several objective functions simultaneously, arises in many areas. Since the objectives are usually conflicting, there does not exist a single solution that can perform well on all these objectives. Thus, the goal of multi-objective optimization is to find a set of Pareto optimal solutions (or the Pareto front, i.e., the set of objective vectors of

* This work was supported by the NSFC (62022039) and the Jiangsu NSF (BK20201247). Chao Qian is the corresponding author. Due to space limitation, proof details are available at https://arxiv.org/abs/2203.11550.

the Pareto optimal solutions), representing different optimal trade-offs between these objectives. Evolutionary algorithms (EAs) [2] are a kind of randomized heuristic optimization algorithms, inspired by natural evolution. They maintain a set of solutions, i.e., a population, and iteratively improve the population by reproducing new solutions and selecting better ones. Due to the population-based nature, EAs are very popular for solving multi-objective optimization problems, and have been widely used in many real-world applications [4].

Compared with practical applications, the theoretical foundation of EAs is still underdeveloped, which is mainly because the sophisticated behaviors of EAs make theoretical analysis quite difficult. Though much effort has been devoted to the essential theoretical aspect, i.e., running time analysis, leading to a lot of progress [1,10,28,34] in the past 25 years, most of them focused on single-objective optimization, while only a few considered the more complicated scenario of multi-objective optimization. In the following, we briefly review the results of running time analyses on multi-objective EAs (MOEAs).

The running time analysis of MOEAs started from GSEMO, a simple MOEA which employs the bit-wise mutation operator to generate an offspring solution in each iteration and keeps the non-dominated solutions generated-so-far in the population. For GSEMO solving the bi-objective optimization problems LOTZ and COCZ, the expected running time has been proved to be $O(n^3)$ [16] and $O(n^2 \log n)$ [3,30], respectively, where n is the problem size. SEMO is a counterpart of GSEMO, which employs the local mutation operator, one-bit mutation, instead of the global bit-wise mutation operator. Laumanns et al. [21] proved that the expected running time of SEMO solving LOTZ and COCZ are $\Theta(n^3)$ and $O(n^2 \log n)$, respectively. Giel and Lehre [17] considered another bi-objective problem OneMinMax, and proved that both GSEMO and SEMO can solve it in $O(n^2 \log n)$ expected running time. Doerr et al. [9] also proved a lower bound $\Omega(n^2/p)$ for GSEMO solving LOTZ, where $p < n^{-7/4}$ is the mutation rate, i.e., the probability of flipping each bit when performing bit-wise mutation.

Later, the analyses of GSEMO were conducted on multi-objective combinatorial optimization problems. For bi-objective minimum spanning trees (MST), GSEMO was proved to be able to find a 2-approximation of the Pareto front in expected pseudo-polynomial time [24]. For multi-objective shortest paths, a variant of GSEMO can achieve an $(1+\epsilon)$-approximation in expected pseudo-polynomial time [18,26], where $\epsilon > 0$. Laumanns et al. [20] considered GSEMO and its variant for solving a special case of the multi-objective knapsack problem, and proved that the expected running time of the two algorithms for finding all the Pareto optimal solutions are $O(n^6)$ and $O(n^5)$, respectively.

There are also studies that analyze GSEMO for solving single-objective constrained optimization problems. By optimizing a reformulated bi-objective optimization problem that optimizes the original objective and a constraint-related objective simultaneously, GSEMO can reduce the expected running time significantly for achieving a desired approximation ratio. For example, by reformulating the set cover problem into a bi-objective problem, Friedrich et al. [12] proved that GSEMO and SEMO can solve a class of set cover instances in

$O(mn(\log c_{\max} + \log n))$ expected running time, which is better than the exponential expected running time of (1+1)-EA, i.e., the single-objective counterpart to GSEMO, where m, n and $c_{\max}$ denote the size of the ground set, the size of the collection of subsets, and the maximum cost of a subset, respectively. More evidence has been proved on the problems of minimum cuts [25], minimum cost coverage [31], MST [27] and submodular optimization [15]. Note that we concern inherently multi-objective optimization problems in this paper.

Based on GSEMO and SEMO, the effectiveness of some strategies for multi-objective evolutionary optimization has been analyzed. For example, Laumanns et al. [21] showed the effectiveness of greedy selection by proving that using this strategy can reduce the expected running time of SEMO from $O(n^2 \log n)$ to $\Theta(n^2)$ for solving the COCZ problem. Qian et al. [30] showed that crossover can accelerate filling the Pareto front by comparing the expected running time of GSEMO with and without crossover for solving the artificial problems COCZ and weighted LPTNO (a generalization of LOTZ), as well as the combinatorial problem multi-objective MST. The effectiveness of some other mechanisms, e.g., heuristic selection [29], diversity [13], fairness [14,21], and diversity-based parent selection [6] have also been examined.

Though GSEMO and SEMO share the general structure of MOEAs, they have been much simplified. To characterize the behavior of practical MOEAs, some efforts have been devoted to analyzing MOEA/D, which is a popular MOEA based on decomposition [32]. Li et al. [23] analyzed a simplified variant of MOEA/D without crossover for solving COCZ and weighted LPTNO, and proved that the expected running time is $\Theta(n \log n)$ and $\Theta(n^2)$, respectively. Huang et al. [19] also considered a simplified MOEA/D, and examined the effectiveness of different decomposition approaches by comparing the running time for solving two many-objective problems mLOTZ and mCOCZ, where m denotes the number of objectives.

Surprisingly, the running time analysis of the non-dominated sorting genetic algorithm II (NSGA-II) [8], the probably most influential MOEA, has been rarely touched. The NSGA-II enables to find well-spread Pareto-optimal solutions by incorporating two substantial features, i.e., non-dominated sorting and crowding distance, and has become the most popular MOEA for solving multi-objective optimization problems [7]. To the best of our knowledge, the only attempt is a very recent work, which, however, considered a simplified version of NSGA-II without crossover, and proved that the expected running time is $O(n^2 \log n)$ for OneMinMax and $O(n^3)$ for LOTZ [33].

In this paper, we present a running time analysis for the standard NSGA-II. We prove that the expected running time of NSGA-II is $O(n^3)$ for solving LOTZ. Note that the running time upper bound is the same as that of GSEMO and SEMO [16,17,21,30], implying that the NSGA-II does not have an advantage over simplified MOEAs on LOTZ if the derived upper bound is tight.

Next, we introduce a new parent selection strategy, i.e., stochastic tournament selection, which samples a number k uniformly at random and then performs k tournament selection. By replacing the original binary tournament

selection of NSGA-II with stochastic tournament selection, we prove that the expected running time of NSGA-II can be improved to $O(n^2)$ for LOTZ. We also conduct experiments, suggesting that the derived upper bounds are tight. Furthermore, we empirically examine the performance of the NSGA-II using the two selection strategies on the widely used benchmark problem ZDT1 [35]. The results show that stochastic tournament selection can help the NSGA-II converge faster, disclosing its potential in practical applications.

2 Preliminaries

In this section, we first introduce multi-objective optimization, and then introduce the procedure of NSGA-II.

2.1 Multi-objective Optimization

Multi-objective optimization requires to simultaneously optimize two or more objective functions, as shown in Definition 1. We consider maximization here, while minimization can be defined similarly. The objectives are usually conflicting, and thus there is no canonical complete order in 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 objective vector in the Pareto front.

Definition 1 (Multi-objective Optimization). *Given a feasible solution space $\mathcal{X}$ and objective functions $f_1, f_2, \ldots, f_m$, 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 $\boldsymbol{y} \in \mathcal{X}$:*

- $\boldsymbol{x}$ weakly dominates $\boldsymbol{y}$ *(denoted as $\boldsymbol{x} \succeq \boldsymbol{y}$) if $\forall 1 \le i \le m, f_i(\boldsymbol{x}) \ge f_i(\boldsymbol{y})$;*
- $\boldsymbol{x}$ dominates $\boldsymbol{y}$ *(denoted as $\boldsymbol{x} \succ \boldsymbol{y}$) if $\boldsymbol{x} \succeq \boldsymbol{y}$ and $f_i(\boldsymbol{x}) > f_i(\boldsymbol{y})$ for some i;*
- $\boldsymbol{x}$ *and* $\boldsymbol{y}$ *are* incomparable *if neither $\boldsymbol{x} \succeq \boldsymbol{y}$ nor $\boldsymbol{y} \succeq \boldsymbol{x}$.*

2.2 NSGA-II

The NSGA-II algorithm [8] as presented in Algorithm 1 is a popular MOEA, which incorporates two substantial features, i.e., non-dominated sorting and crowding distance. NSGA-II starts from an initial population of N random solutions (line 1). In each generation, it employs binary tournament selection N times to generate a parent population P' (line 4), and then applies one-point crossover and bit-wise mutation on the $N/2$ pairs of parent solutions to generate N offspring solutions (lines 5–9). Note that the two adjacent selected solutions

Algorithm 1 NSGA-II Algorithm [8]

Input: objective functions $f_1, f_2 \ldots, f_m$, population size N
Output: N solutions from $\{0,1\}^n$

1: $P \leftarrow N$ solutions uniformly and randomly selected from $\{0,1\}^n$;
2: **while** criterion is not met **do**
3: $Q = \emptyset$;
4: apply binary tournament selection N times to generate a parent population P' of size N;
5: **for** each consecutive pair of the parent solutions $\boldsymbol{x}$ and $\boldsymbol{y}$ in P' **do**
6: apply one-point crossover on $\boldsymbol{x}$ and $\boldsymbol{y}$ to generate two solutions $\boldsymbol{x}'$ and $\boldsymbol{y}'$, with probability 0.9;
7: apply bit-wise mutation on $\boldsymbol{x}'$ and $\boldsymbol{y}'$ to generate $\boldsymbol{x}''$ and $\boldsymbol{y}''$, respectively;
8: add $\boldsymbol{x}''$ and $\boldsymbol{y}''$ into Q
9: **end for**
10: partition $P \cup Q$ into non-dominated sets $F_1, F_2, \ldots$;
11: let $P = \emptyset$, $i = 1$;
12: **while** $|P \cup F_i| < N$ **do**
13: $P = P \cup F_i$, $i = i + 1$
14: **end while**
15: assign each solution in F_i with a crowding distance;
16: sort the solutions in F_i by crowding distance in descending order, and add the first $N - |P|$ solutions into P
17: **end while**
18: **return** P

form a pair, and thus the N selected solutions form $N/2$ pairs. The one-point crossover operator first selects a crossover point $i \in \{1, 2, \ldots, n\}$ uniformly at random, where n is the problem size, and then exchanges the first i bits of two solutions. The bit-wise mutation operator flips each bit of a solution independently with probability $1/n$. Note that for real-coded solutions (which, however, are not considered in this paper), the one-point crossover operator and bit-wise mutation operator can be replaced by other operators, e.g., the simulated binary crossover (SBX) operator and polynomial mutation operator [8]. The binary tournament selection presented in Definition 3 picks two solutions randomly from the population P with or without replacement, and then selects a better one (ties broken uniformly). Note that we consider the strategy with replacement in this paper.

Definition 3 (Binary Tournament Selection). *The binary tournament selection strategy first picks two solutions from the population P uniformly at random, and then selects a better one with ties broken uniformly.*

After generating N offspring solutions, the best N solutions in the current population P and the offspring population Q are selected as the population in the next generation (lines 10–16). In particular, the solutions in the current and offspring populations are partitioned into non-dominated sets $F_1, F_2, \ldots$ (line 10), where F_1 contains all the non-dominated solutions in $P \cup Q$, and F_i ($i \geq 2$) contains all the non-dominated solutions in $(P \cup Q) \setminus \cup_{j=1}^{i-1} F_j$. Note that we use

the notion $\texttt{rank}(\boldsymbol{x}) = i$ to denote that $\boldsymbol{x}$ belongs to F_i. Then, the solutions in $F_1, F_2, \ldots$ are added into the next population (lines 12–14), until the population size exceeds N. For the critical set F_i whose inclusion makes the population size larger than N, the crowding distance is computed for each of the contained solutions (line 15). Finally, the solutions in F_i with large crowding distance are selected to fill the remaining population slots (line 16).

When using binary tournament selection (line 4), the selection criterion is based on the crowded-comparison, i.e., $\boldsymbol{x}$ is superior to $\boldsymbol{y}$ (denoted as $\boldsymbol{x} \succ_{\rm c} \boldsymbol{y}$) if

$$\texttt{rank}(\boldsymbol{x}) < \texttt{rank}(\boldsymbol{y}) \;\; \text{OR} \;\; \texttt{rank}(\boldsymbol{x}) = \texttt{rank}(\boldsymbol{y}) \wedge \texttt{dist}(\boldsymbol{x}) > \texttt{dist}(\boldsymbol{y}). \quad (1)$$

Intuitively, the crowding distance of a solution means the distance between its closest neighbour solutions, and a solution with larger crowding distance is preferred so that the diversity of the population can be preserved as much as possible. Note that when computing the crowding distance, we assume that the relative positions of the solutions with the same objective vector are unchanged or totally reversed when the solutions are sorted w.r.t. some objective function. Such requirement can be met by any *stable* sorting algorithm, e.g., the bubble sort or merge sort, which maintains the relative order of items with equal keys (i.e., values). What's more, the built-in sorting functions in MATLAB, e.g., sortrows() and sort(), can also satisfy the requirement. Under such assumption, the population size needed to find the Pareto front can be reduced (a detailed discussion is provided after Theorem 1).

In line 6 of Algorithm 1, the probability of using crossover has been set to 0.9, which is the same as the original setting and also commonly used [8]. However, the theoretical results derived in this paper can be directly generalized to the scenario where the probability of using crossover belongs to $[\Omega(1), 1 - \Omega(1)]$.

3 Running Time Analysis of NSGA-II

In this section, we analyze the expected running time of the standard NSGA-II in Algorithm 1 solving the bi-objective pseudo-Boolean problem LOTZ, which is widely used in MOEAs' theoretical analyses [9,21,30].

The LOTZ problem presented in Definition 4 aims to maximize the number of leading 1-bits and the number of trailing 0-bits of a binary bit string. The Pareto front of LOTZ is $\mathcal{F} = \{(0, n), (1, n-1), \ldots, (n, 0)\}$, and the corresponding Pareto optimal solutions are $0^n, 10^{n-1}, \ldots, 1^n$.

Definition 4 (LOTZ [21]). *The LOTZ problem of size n is to find n bits binary strings which maximize $\boldsymbol{f}(\boldsymbol{x}) = \left(\sum_{i=1}^{n} \prod_{j=1}^{i} x_j, \sum_{i=1}^{n} \prod_{j=i}^{n} (1 - x_j)\right)$, where x_j denotes the j-th bit of $\boldsymbol{x} \in \{0,1\}^n$.*

We prove in Theorem 1 that the NSGA-II can find the Pareto front in $O(n^2)$ expected number of generations, i.e., $O(n^3)$ expected number of fitness evaluations, because the generated N offspring solutions need to be evaluated in each iteration. Note that the running time of an EA is usually measured by the number of fitness evaluations, because evaluating the fitness of a solution is often the

most time-consuming step in practice. The main proof idea can be summarized as follows. The NSGA-II first employs the mutation operator to find the two solutions with the largest number of leading 1-bits and the largest number of trailing 0-bits, i.e., 1^n and 0^n, respectively; then employs the crossover operator to find the whole Pareto front.

Theorem 1. *For the NSGA-II solving LOTZ, if using binary tournament selection and a population size N such that $2n+2 \leq N = O(n)$, then the expected number of generations for finding the Pareto front is $O(n^2)$.*

Note that Zheng et al. [33] proved that for NSGA-II using bit-wise mutation (without crossover), the expected running time is $O(Nn^2)$ if the population size N is at least $5n+5$. Thus, the requirement for the population size N is relaxed from $5n+5$ to $2n+2$. The main reason for the relaxation is that under the assumption in Section 2.2 (i.e., the order of the solutions with the same objective vector is unchanged or totally reversed when the solutions are sorted according to some f_j), there exist at most two solutions with i leading 1-bits such that their ranks are equal to 1 and crowding distances are larger than 0, for each $i \in \{0, 1, \ldots, n\}$. Meanwhile, for any objective vector in the Pareto front that has been obtained by the algorithm, there is at least one corresponding solution in the population such that its rank is equal to 1 and crowding distance is larger than 0, implying that the solution will occupy one of the $2n+2$ slots, and thus be maintained in the population. Without such assumption, there may exist more solutions with crowding distance larger than 0 for each objective vector in the Pareto front, thus requiring a larger population size.

4 NSGA-II Using Stochastic Tournament Selection

In the previous section, we have proved that the expected running time of the standard NSGA-II is $O(n^3)$ for LOTZ, which is the same as that of the previously analyzed simple MOEAs, GSEMO and SEMO [21,30]. Next, we introduce a new parent selection strategy, i.e., the stochastic tournament selection, into the NSGA-II, and show that the expected running time needed to find the whole Pareto front can be reduced to $O(n^2)$.

4.1 Stochastic Tournament Selection

As the crowded-comparison $\succ_c$ in Eq. (1) actually gives a total order of the solutions in the population P, binary tournament selection can be naturally extended to k tournament selection [11], as presented in Definition 5, where k is a parameter such that $1 \leq k \leq N$. That is, k solutions are first picked from P uniformly at random, and then the solution with the smallest rank is selected. If several solutions have the same smallest rank, the one with the largest crowding distance is selected, with ties broken uniformly.

Definition 5 (k Tournament Selection). *The k tournament selection strategy first picks k solutions from the population P uniformly at random, and then selects the best one with ties broken uniformly.*

Note that a larger k implies a larger selection pressure, i.e., a larger probability of selecting a good solution, and thus the value of k can be used to control the selection pressure of EAs [11]. However, this also brings about a new issue, i.e., how to set k properly. In order to reduce the risk of setting improper values of k as well as the overhead of tuning k, we introduce a natural strategy, i.e., stochastic tournament selection in Definition 6, which first selects a number k randomly, and then performs the k tournament selection. In this paper, we consider that the tournament candidates are picked with replacement.

Definition 6 (Stochastic Tournament Selection). *The stochastic tournament selection strategy first selects a number k from $\{1, 2, \ldots, N\}$ uniformly at random, where N is the size of the population P, and then employs the k tournament selection to select a solution from the population P.*

In each generation of NSGA-II, we need to select N parent solutions independently, and each selection may involve the comparison of several solutions, which may lead to a large number of comparisons. To improve the efficiency of stochastic tournament selection, we can first sort the solutions in the population P, and then perform the parent selection procedure. Specifically, each solution $\boldsymbol{x}_i$ ($1 \leq i \leq N$) in P is assigned a number $\pi(i)$, where $\pi : \{1, 2, \ldots, N\} \to \{1, 2, \ldots, N\}$ is a bijection such that

$$\forall 1 \leq i, j \leq N, i \neq j : \boldsymbol{x}_i \succ_{\mathrm{c}} \boldsymbol{x}_j \Rightarrow \pi(i) < \pi(j). \tag{2}$$

That is, a solution with a smaller number is better. Note that the number $\pi(\cdot)$ is assigned randomly if several solutions have the same rank and crowding distance. Then, we sample a number k randomly from $\{1, 2, \ldots, N\}$ and pick k solutions from P at random, where the solution with the lowest $\pi(\cdot)$ value is finally selected.

Lemma 1 presents the property of stochastic tournament selection, which will be used in the following theoretical analysis. It shows that any solution (even the worst solution) in P can be selected with probability at least $1/N^2$, and any solution belonging to the best $O(1)$ solutions in P (with respect to $\succ_{\mathrm{c}}$) can be selected with probability at least $\Omega(1)$. Note that for binary tournament selection, the probability of selecting the worst solution (denoted as $\boldsymbol{x}^{\mathrm{w}}$) is $1/N^2$, because $\boldsymbol{x}^{\mathrm{w}}$ is selected if and only if the two solutions picked for competition are both $\boldsymbol{x}^{\mathrm{w}}$; the probability of selecting the best solution (denoted as $\boldsymbol{x}^{\mathrm{b}}$) is $1 - (1 - 1/N)^2 = 2/N - 1/N^2$, because $\boldsymbol{x}^{\mathrm{b}}$ is selected if and only if $\boldsymbol{x}^{\mathrm{b}}$ is picked at least once. Thus, compared with binary tournament selection, stochastic tournament selection can increase the probability of selecting the top solutions, and meanwhile maintain the probability of selecting the bottom solutions. Note that such probability is similar to the power law distribution in [6].

Lemma 1. *If using stochastic tournament selection, any solution in P can be selected with prob. at least $1/N^2$. Furthermore, a solution $\boldsymbol{x}_i \in P$ with $\pi(i) = O(1)$ can be selected with prob. $\Omega(1)$, where $\pi : \{1, 2, \ldots, N\} \to \{1, 2, \ldots, N\}$ is a bijection satisfying Eq.* (2).

4.2 Running Time Analysis

We prove that the expected number of generations of the NSGA-II using stochastic tournament selection is $O(n)$ (implying $O(n^2)$ expected running time) for solving LOTZ, in Theorem 2. The proof idea of Theorem 2 is similar to that of Theorem 1. That is, the NSGA-II first employs the mutation operator to find the solutions that maximize each objective function, and then employs the crossover operator to quickly find the remaining objective vectors in the Pareto front. However, the utilization of stochastic tournament selection can make the NSGA-II select prominent solutions, i.e., solutions maximizing each objective function, with larger probability, making the crossover operator easier fill in the remaining Pareto front and thus reducing the total running time.

Theorem 2. *For the NSGA-II solving LOTZ, if using stochastic tournament selection and a population size N such that $2n + 2 \leq N = O(n)$, then the expected number of generations for finding the Pareto front is $O(n)$.*

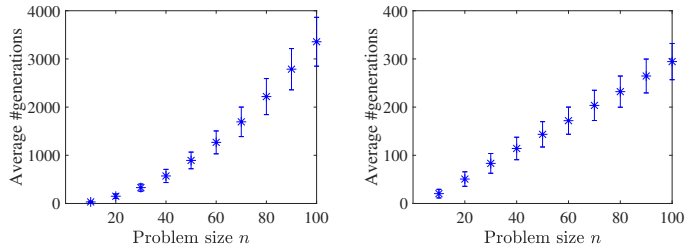
5 Experiments

In the previous sections, we have proved that when binary tournament selection is used in the NSGA-II, the expected number of generations is $O(n^2)$ for LOTZ; when stochastic tournament selection is used, the expected number of generations can be improved to $O(n)$. But as the lower bounds on the running time have not been derived, the comparison may be not strict. Thus, we conduct experiments to examine the tightness of these upper bounds. We also conduct experiments on the widely used benchmark problem ZDT1 [35], to examine the performance of the stochastic tournament selection in more realistic scenarios.

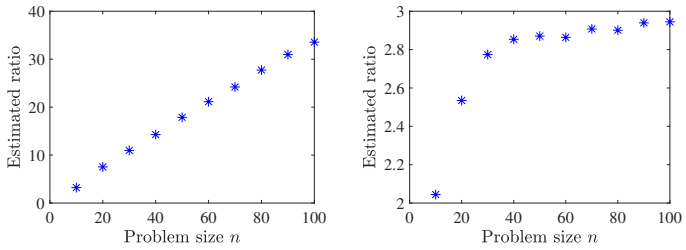
5.1 LOTZ Problem

For the LOTZ problem, we examine the performance of NSGA-II when the problem size n changes from 10 to 100, with a step of 10. On each problem size n, we run the NSGA-II 1000 times independently, and record the number of generations until the Pareto front is found. Then, the average number of generations and the standard deviation of the 1000 runs are reported in Figure 1(a). To show the relationship between the average number of generations and the problem size n clearly, we also plot the estimated ratio, i.e., the average number of generations divided by the problem size n, in Figure 1(b).

From the left subfigures of Figure 1(a) and 1(b), we can observe that the average number of generations is approximately $\Theta(n^2)$, suggesting that the upper bound $O(n^2)$ derived in Theorem 1 is tight. By the right subfigures of Figure 1(a) and 1(b), the average number of generations is clearly a linear function of n, which suggests that the upper bound $O(n)$ derived in Theorem 2 is also tight.



(a) Average #generations



(b) Average #generations divided by the problem size n

Fig. 1. Average #generations and the estimated ratio of the NSGA-II using binary tournament selection or stochastic tournament selection for solving the LOTZ problem. Left subfigure: the NSGA-II using binary tournament selection; right subfigure: the NSGA-II using stochastic tournament selection.

5.2 ZDT1 Problem

The ZDT1 problem presented in Definition 7 is a widely used benchmark to test the practical performance of MOEAs [35]. It has 30 continuous decision variables with each variable taking value from $[0, 1]$. As suggested in [8], we use 30 bits (i.e., a binary string of length 30) to code each decision variable. The Pareto front of ZDT1 is $\mathcal{F} = \{(f_1, 1 - \sqrt{f_1}) \mid f_1 \in [0, 1]\}$. Note that ZDT1 is a minimization problem, and thus we need to change the domination relationship in Definition 2 from "$f_i(\boldsymbol{x}) \geq (\text{or} >) f_i(\boldsymbol{y})$" to "$f_i(\boldsymbol{x}) \leq (\text{or} <) f_i(\boldsymbol{y})$" accordingly.

Definition 7 (ZDT1 [35]). *The ZDT1 problem is to find a 30-dimensional decision vector* $\boldsymbol{x} = (x_1, x_2, \ldots, x_{30})$ *which minimizes* $\boldsymbol{f}(\boldsymbol{x}) = \big(x_1, g(\boldsymbol{x})\big(1 - \sqrt{x_1/g(\boldsymbol{x})}\ \big)\big)$*, where* $\forall 1 \leq i \leq 30 : x_i \in [0, 1]$*, and* $g(\boldsymbol{x}) = 1 + 9 \cdot \sum_{i=2}^{30} x_i/29$.

Different from the LOTZ problem, the Pareto front of the ZDT1 problem is an uncountable set. Thus, instead of examining the running time for finding the whole Pareto front, we run NSGA-II for a fixed number of generations, i.e., 300, and examine the quality of the obtained population. To measure the quality of a set of solutions, we use the inverted generational distance (IGD) indicator, which has been widely used in multi-objective optimization [5,22]. As presented in Definition 8, $\text{IGD}(R, A)$ intuitively means the average distance of the reference points in R to the objective vectors in A, where the reference points

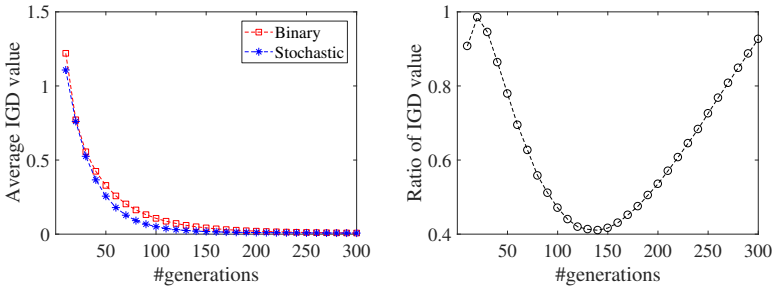


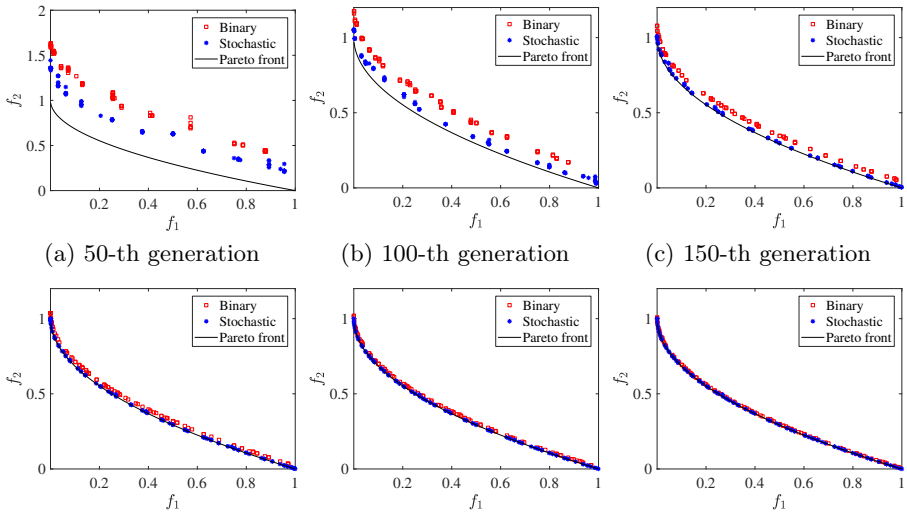
Fig. 2. Average IGD value of the NSGA-II using binary or stochastic tournament selection for solving the ZDT1 problem. Left subfigure: average IGD value of the NSGA-II vs. #generations; right subfigure: average IGD value of the NSGA-II using stochastic tournament selection divided by that of the NSGA-II using binary tournament selection vs. #generations.

are usually sampled from the Pareto front in advance, and the set A consists of objective vectors of the solutions in the population. It is straightforward to see that a smaller IGD value implies a better approximation of the population to the Pareto front, in terms of both convergence and diversity.

Definition 8 (IGD [5]). *Given a set $R = \{\boldsymbol{r}^1, \boldsymbol{r}^2, \ldots, \boldsymbol{r}^l\}$ of reference points and a set A of objective vectors, the IGD value of the set A with respect to R is defined as $IGD(R, A) = \frac{1}{l}\sum_{i=1}^{l} \min_{\boldsymbol{a}\in A} d_2(\boldsymbol{r}^i, \boldsymbol{a})$, where $d_2(\boldsymbol{r}^i, \boldsymbol{a})$ denotes the Euclidean distance between $\boldsymbol{r}^i$ and $\boldsymbol{a}$.*

In our experiments, we sample 200 points uniformly from the Pareto front as the reference set R, and set the population size N to 100. We run the NSGA-II 1000 times independently, and report the average IGD value of the 1000 runs every 10 generations. From Figure 2(a), we can observe that (i) initially, the two selection strategies achieve similar performance; (ii) in the intermediate stage, the NSGA-II using stochastic tournament selection converges to the Pareto front faster than the NSGA-II using binary tournament selection; (iii) finally, the two selection strategies both achieve IGD value very close to 0, implying a good approximation ability of the two strategies. We also plot the ratio of the average IGD value obtained by the NSGA-II using stochastic tournament selection and binary tournament selection in Figure 2(b). We can observe that the ratio is always at most 1, and decreases rapidly in the initial optimization procedure, implying that stochastic tournament selection is always better, and can help the NSGA-II converge faster. As time goes by, the advantage of stochastic tournament selection diminishes, because the NSGA-II using the two strategies have both found objective vectors which approximate the Pareto front well.

To better visualize the performance of the NSGA-II using the two selection strategies, we also plot the objective vectors obtained by the NSGA-II every 50 generations in one of the runs. Figure 3 shows that the objective vectors obtained by the NSGA-II using stochastic tournament selection are always evenly



(a) 50-th generation (b) 100-th generation (c) 150-th generation

(d) 200-th generation (e) 250-th generation (f) 300-th generation

Fig. 3. Objective vectors obtained by the NSGA-II using binary or stochastic tournament selection for solving the ZDT1 problem.

distributed along the Pareto front, and gradually converge, suggesting the good spread ability of stochastic tournament selection.

In summary, the two selection strategies can achieve similar performance when given long enough time, but stochastic tournament selection can help the NSGA-II converge faster. The reason may be that the second objective value of the ZDT1 problem can be consecutively decreased by decreasing the value of $x_2, x_3, \ldots, x_{30}$, i.e., a currently good solution is helpful in the subsequent optimization process; and stochastic tournament selection can take advantage of these good solutions more efficiently.

6 Conclusion

In this paper, we theoretically analyze the running time of the NSGA-II solving the bi-objective problem LOTZ, and derive the upper bound that is the same as that of the previously analyzed simple MOEAs, GSEMO and SEMO. Then, we propose a new parent selection strategy, stochastic tournament selection, to replace the binary tournament selection strategy of the NSGA-II, and prove that the NSGA-II using the new strategy can find the Pareto front of LOTZ with a much smaller running time upper bound. Experimental results suggest that the derived upper bounds on LOTZ are tight, and also show the superior performance of stochastic tournament selection on the widely used benchmark problem ZDT1. In the future, we will analyze the lower bounds on the running time to make the comparison strict, and it is also interesting to examine the effectiveness of stochastic tournament selection on more problems.

References

1. Auger, A., Doerr, B.: Theory of Randomized Search Heuristics: Foundations and Recent Developments. World Scientific, Singapore (2011)
2. Bäck, T.: Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms. Oxford University Press, Oxford, UK (1996)
3. Bian, C., Qian, C., Tang, K.: A general approach to running time analysis of multi-objective evolutionary algorithms. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI'18). pp. 1405–1411. Stockholm, Sweden (2018)
4. Coello Coello, C.A., Lamont, G.B.: Applications of Multi-Objective Evolutionary Algorithms. World Scientific, Singapore (2004)
5. Coello Coello, C.A., Sierra, M.R.: A general approach to running time analysis of multi-objective evolutionary algorithms. In: Proceedings of the Mexican International Conference on Artificial Intelligence (MICAI'04). pp. 688–697. Mexico City, Mexico (2004)
6. Covantes Osuna, E., Gao, W., Neumann, F., Sudholt, D.: Design and analysis of diversity-based parent selection schemes for speeding up evolutionary multi-objective optimisation. Theoretical Computer Science **832**, 123–142 (2020)
7. Deb, K.: Multi-objective optimisation using evolutionary algorithms: an introduction. In: Multi-objective Evolutionary Optimisation for Product Design and Manufacturing, pp. 3–34. Springer (2011)
8. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6**(2), 182–197 (2002)
9. Doerr, B., Kodric, B., Voigt, M.: Lower bounds for the runtime of a global multi-objective evolutionary algorithm. In: Proceedings of the 2013 IEEE Congress on Evolutionary Computation (CEC'13). pp. 432–439. Cancun, Mexico (2013)
10. Doerr, B., Neumann, F. (eds.): Theory of Evolutionary Computation: Recent Developments in Discrete Optimization. Natural Computing Series, Springer (2020)
11. E. Eiben, A., E. Smith, J.: Introduction to Evolutionary Computing. Springer-Verlag, Berlin, Germany (2015)
12. Friedrich, T., He, J., Hebbinghaus, N., Neumann, F., Witt, C.: Approximating covering problems by randomized search heuristics using multi-objective models. Evolutionary Computation **18**(4), 617–633 (2010)
13. Friedrich, T., Hebbinghaus, N., Neumann, F.: Plateaus can be harder in multi-objective optimization. Theoretical Computer Science **411**(6), 854–864 (2010)
14. Friedrich, T., Horoba, C., Neumann, F.: Illustration of fairness in evolutionary multi-objective optimization. Theoretical Computer Science **412**(17), 1546–1556 (2011)
15. Friedrich, T., Neumann, F.: Maximizing submodular functions under matroid constraints by evolutionary algorithms. Evolutionary Computation **23**(4), 543–558 (2015)
16. Giel, O.: Expected runtimes of a simple multi-objective evolutionary algorithm. In: Proceedings of the 2003 IEEE Congress on Evolutionary Computation (CEC'03). pp. 1918–1925. Canberra, Australia (2003)
17. Giel, O., Lehre, P.K.: On the effect of populations in evolutionary multi-objective optimisation. Evolutionary Computation **18**(3), 335–356 (2010)

18. Horoba, C.: Analysis of a simple evolutionary algorithm for the multiobjective shortest path problem. In: Proceedings of the 10th International Workshop on Foundations of Genetic Algorithms (FOGA'09). pp. 113–120. Orlando, FL (2009)
19. Huang, Z., Zhou, Y., Luo, C., Lin, Q.: A runtime analysis of typical decomposition approaches in MOEA/D framework for many-objective optimization problems. In: Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI'21). pp. 1682–1688. Virtual (2021)
20. Laumanns, M., Thiele, L., Zitzler, E.: Running time analysis of evolutionary algorithms on a simplified multiobjective knapsack problem. Natural Computing **3**, 37–51 (2004)
21. 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)
22. Li, M., Yao, X.: Quality evaluation of solution sets in multiobjective optimisation: A survey. ACM Computing Surveys **52**(2), 26:1–38 (2020)
23. Li, Y., Zhou, Y., Zhan, Z., Zhang, J.: A primary theoretical study on decomposition-based multiobjective evolutionary algorithms. IEEE Transactions on Evolutionary Computation **20**(4), 563–576 (2016)
24. Neumann, F.: Expected runtimes of a simple evolutionary algorithm for the multiobjective minimum spanning tree problem. European Journal of Operational Research **181**(3), 1620–1629 (2007)
25. Neumann, F., Reichel, J., Skutella, M.: Computing minimum cuts by randomized search heuristics. Algorithmica **59**, 323–342 (2011)
26. Neumann, F., Theile, M.: 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). pp. 667–676. Krakow, Poland (2010)
27. Neumann, F., Wegener, I.: Minimum spanning trees made easier via multi-objective optimization. Natural Computing **5**, 305–319 (2006)
28. Neumann, F., Witt, C.: Bioinspired Computation in Combinatorial Optimization: Algorithms and Their Computational Complexity. Springer-Verlag, Berlin, Germany (2010)
29. Qian, C., Tang, K., Zhou, Z.H.: 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). pp. 835–846. Edinburgh, Scotland (2016)
30. Qian, C., Yu, Y., Zhou, Z.H.: An analysis on recombination in multi-objective evolutionary optimization. Artificial Intelligence **204**, 99–119 (2013)
31. Qian, C., Yu, Y., Zhou, Z.H.: On constrained Boolean Pareto optimization. In: Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI'15). pp. 389–395. Buenos Aires, Argentina (2015)
32. Zhang, Q., Li, H.: MOEA/D: A multiobjective evolutionary algorithm based on decomposition. IEEE Transactions on Evolutionary Computation **11**(6), 712–731 (2007)
33. Zheng, W., Liu, Y., Doerr, B.: A first mathematical runtime analysis of the non-dominated sorting genetic algorithm II (NSGA-II). In: Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI'22). to appear. Virtual (2022)
34. Zhou, Z.H., Yu, Y., Qian, C.: Evolutionary Learning: Advances in Theories and Algorithms. Springer, Singapore (2019)
35. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary Computation **8**(2), 173–195 (2000)