How to Use the Metropolis Algorithm for Multi-Objective Optimization?

Weijie Zheng[1], **Mingfeng Li**[1], **Renzhong Deng**[1], **Benjamin Doerr**[2*]

[1] School of Computer Science and Technology, International Research Institute for Artificial Intelligence, Harbin Institute of Technology, Shenzhen, China

[2] Laboratoire d'Informatique (LIX), École Polytechnique, CNRS, Institut Polytechnique de Paris, Palaiseau, France

zhengweijie@hit.edu.cn, lastname@lix.polytechnique.fr

Abstract

The Metropolis algorithm can cope with local optima by accepting inferior solutions with suitably small probability. That this can work well was not only observed in empirical research, but also via mathematical runtime analyses on single-objective benchmarks. This paper takes several steps towards understanding, again via theoretical means, whether such advantages can also be obtained in multi-objective optimization.

The original Metropolis algorithm has two components, one-bit mutation and the acceptance strategy, which allows accepting inferior solutions. When adjusting the acceptance strategy to multi-objective optimization in the way that an inferior solution that is accepted replaces its parent, then the Metropolis algorithm is not very efficient on our multi-objective version of the multimodal DLB benchmark called DLTB. With one-bit mutation, this multi-objective Metropolis algorithm cannot optimize the DLTB problem, with standard bit-wise mutation it needs at least $\Omega(n^5)$ time to cover the full Pareto front. In contrast, we show that many other multi-objective optimizers, namely the GSEMO, SMS-EMOA, and NSGA-II, only need time $O(n^4)$.

When keeping the parent when an inferior point is accepted, the multi-objective Metropolis algorithm both with one-bit or standard bit-wise mutation solves the DLTB problem efficiently, with one-bit mutation experimentally leading to better results than several other algorithms.

Overall, our work suggests that the general mechanism of the Metropolis algorithm can be interesting in multi-objective optimization, but that the implementation details can have a huge impact on the performance.

Introduction

Simulated annealing is widely applied in both single- and multi-objective optimization problems, see (Suman and Kumar 2006). The Metropolis algorithm (Metropolis et al. 1953) is a special case of simulated annealing with a fixed temperature. The runtime analysis of the Metropolis algorithm for single-objective optimization started in 1988, when Sasaki and Hajek (1988) proved that good approximations for the maximum matching problem can be computed in polynomial time. Jansen and Wegener (2007) (see Doerr et al. (2023) for a recent tightening of this result) proved that the Metropolis algorithm with suitable parameters solves the ONEMAX benchmark in an expected time $O(n \log n)$ fitness evaluation, a runtime many other randomized search heuristics have as well on this problem (Droste, Jansen, and Wegener 2002; Jansen, Jong, and Wegener 2005; Witt 2006; Sudholt and Witt 2019; Doerr and Krejca 2020; Antipov and Doerr 2021).

The Metropolis algorithm can cope with local optima by accepting inferior solutions with suitably small probability. Jansen and Wegener (2007) proved that the Metropolis algorithm solves the GENTLENEGATIVESLOPE problem in polynomial time, whereas the $(1 + 1)$ EA needs at least exponential time; that also the move acceptance hyper-heuristic need super-polynomial time, was recently shown by Lissovoi, Oliveto, and Warwicker (2023). Oliveto et al. (2018) proved that the Metropolis algorithm solves their VALLEY function (which is different from the VALLEY problem defined in (Droste, Jansen, and Wegener 2000)) more efficiently than simple evolutionary algorithms (EAs). Wang, Zheng, and Doerr (2024) showed that the Metropolis algorithm solves the DECEPTIVELEADINGBLOCKS (DLB) function in expected time of $O(n^2)$, while all (1+1)-elitist unary unbiased black-box algorithms have an expected runtime of $\Omega(n^3)$.

We note that the known runtime results for the Metropolis algorithm are all for single optimization. In this paper, we take several steps towards theoretically analyzing multi-objective Metropolis algorithms. Since an advantage of the Metropolis algorithm was proven for the single-objective DLB function (Wang, Zheng, and Doerr 2024), we construct a bi-objective counterpart of the DLB problem following a general construction method from (Laumanns, Thiele, and Zitzler 2004). We call our new benchmark DLTB, for deceptive leading blocks for ones and deceptive trailing blocks for zeros. DLTB is the first bi-objective multimodal function for theoretical analysis where not all local optima are Pareto optimal. It is also the first bi-objective function in the theory community where the maximum size of a set of mutually non-dominating objective values is larger than the Pareto front size.

The original Metropolis algorithm uses two operators, one-bit mutation as variation operator and the selection mechanism that allows also the inferior offspring to enter

*Corresponding author.

the next population. Since the Metropolis algorithm differs from the randomized local search (RLS) algorithm only in the selection operator, in our discussion of multi-objective Metropolis algorithms we will build upon the simple evolutionary multi-objective optimizer (SEMO), the multi-objective analogue of RLS. Since the inferior solution in the single-objective Metropolis algorithm replaces its parent when accepted, a natural way to implement a multi-objective Metropolis algorithms would be to let also here the inferior solution replace its parent (when accepted). However, we will prove that this variant (and also the original SEMO) cannot cover the Pareto front of the DLTB. Hence the advantage of the Metropolis algorithm over RLS on the single-objective DLB problem does not extend to the bi-objective DLTB problem.

The proof of this negative result heavily exploits that as variation operator one-bit flips are used. For this reason, we also consider the multi-objective Metropolis variant where we replace one-bit mutation with the global operator of the standard bit-wise mutation (flipping each bit independently with probability $1/n$). Unfortunately, we can still prove that this variant needs an expected time of $\Omega(n^5)$ to cover the Pareto front of the DLTB problem. We note that the reason for these lower bounds is not an intrinsic difficulty of the DLTB problem. For three well-known multi-objective EAs, namely GSEMO (Giel 2003), NSGA-II (Deb et al. 2002), and SMS-EMOA (Beume, Naujoks, and Emmerich 2007), we prove that they all cover the Pareto front of the DLTB problem in expected time $O(n^4)$.

Given these unfavorable results for the Metropolis algorithm, we analyze the multi-objective Metropolis algorithm that, in case an inferior solution is accepted, only removes all individuals weakly dominated by it, but not its parent. We prove that this variant covers the Pareto front in expected time $O(n^5)$. Speculating that this runtime guarantee is not tight, we conduct a small experimental investigation and observe that this variant with standard bit-wise mutation achieves similar performance as the GSEMO, NSGA-II, and SMS-EMOA (for which we have shown a runtime guarantee of $O(n^4)$). Interestingly, the experimental results suggest an even better performance of this variant with one-bit mutation.

Preliminaries

This paper considers multi-objective optimization on the search space $\{0,1\}^n$. Let $f = (f_1, \dots, f_m) : \{0,1\}^n \to \mathbb{R}^m$ be the m-objective function to be maximized. We say that x *weakly dominates* y, denoted as $x \succeq y$, if $f_i(x) \geq f_i(y)$ holds for all $i \in [1..m]$, where we denote the integer set $\{a, a+1, \dots, b\}$ as $[a..b]$, $a \leq b$, in this paper. If at least one of the inequalities in the definition of weak domination is strict, then we say that x *dominates* y, denoted as $x \succ y$. If x is not dominated by any other solution in $\{0,1\}^n$, we call x a *Pareto optimum*. The set of all Pareto optima is called *Pareto set*, and the set of function values of all Pareto optima is called *Pareto front*. As common in the evolutionary computation theory community (Neumann and Witt 2010; Auger and Doerr 2011; Jansen 2013; Zhou, Yu, and Qian 2019; Doerr and Neumann 2020), we consider as *runtime* the number of function evaluations until we have a population whose function values include the whole Pareto front.

Algorithm 1: RLS to maximize $f : \{0,1\}^n \to \mathbb{R}$

```
Generate a search point x uniformly in {0,1}^n
loop
    Generate x' via flipping one bit of x uniformly at random
    if f(x') ≥ f(x) then
        x ← x'
    end if
end loop
```

Algorithm 2: The Metropolis algorithm with acceptance parameter $\alpha > 1$ to maximize $f : \{0,1\}^n \to \mathbb{R}$

```
Generate a search point x uniformly in {0,1}^n
loop
    Generate x' via flipping one bit of x uniformly at random
    if f(x') ≥ f(x) then
        x ← x'
    else
        Choose b ∈ {0,1} randomly with Pr[b = 1] = α^(f(x')−f(x))
        if b = 1 then
            x ← x'
        end if
    end if
end loop
```

The single-objective Metropolis algorithm and RLS maintain a population consisting of a single individual. They start with a solution selected uniformly at random. In each iteration an offspring is generated via *one-bit mutation*, that is, by flipping one bit value uniformly at random in the current individual (parent). The only difference between the Metropolis algorithm and RLS is the survival selection. For RLS, the offspring replaces its parent only when it has a better or equal function value. The Metropolis algorithm additionally allows the inferior solution to survive, but with a probability exponentially decreasing with the fitness loss. See Algorithms 1 and 2 for the details.

Due to the similarity, a natural way to define a Metropolis algorithm for multiple objectives is to resort to the multi-objective counterpart of RLS, the SEMO (Laumanns, Thiele, and Zitzler 2004). The SEMO is a well-studied benchmark algorithm in multi-objective evolutionary theory. It starts with a randomly generated individual. In each iteration, a parent is randomly selected to generate an offspring via one-bit mutation. If this offspring is not dominated by any individual in the current population, then it will enter the population and remove all individuals that are dominated by it. See Algorithm 3.

Algorithm 3: SEMO to maximize $f : \{0,1\}^n \to \mathbb{R}^m$

1: Generate $x \in \{0,1\}^n$ uniformly at random and $P \leftarrow \{x\}$
2: **loop**
3: Uniformly at random select one individual x from P
4: Generate x_0 via flipping one bit chosen uniformly at random
5: **if** there is no $y \in P$ such that $x_0 \prec y$ **then**
6: $P = \{z \in P \mid z \not\preceq x_0\} \cup \{x_0\}$
7: **end if**
8: **end loop**

Bi-Objective DLB Problem

In this section, we introduce a bi-objective version of the DLB problem. We will show that it has two interesting features that are not seen in the existing multi-objective benchmark problems.

The single-objective DLB problem was defined by Lehre and Nguyen (2019) to analyze how estimation-of-distribution algorithms cope with deceptive fitness landscapes (see (Doerr and Krejca 2021) for a subsequent work with very different findings). For an even integer n, called the *problem size*, an n-bit string $x = (x_1, \dots, x_n) \in \{0,1\}^n$ is divided into $n/2$ blocks, $x_1x_2, x_3x_4, \dots, x_{n-1}x_n$. Let $m \in [0..n/2-1]$ be minimal such that $x_{2m+1}x_{2m+2} \neq 11$. We call such a block the *critical block*. Then the DLB function value of x is $2m+1$ if $x_{2m+1} + x_{2m+2} = 0$ and is $2m$ if $x_{2m+1} + x_{2m+2} = 1$. The objective value of x is n if x does not possess a critical block, that is, if $x = (1, \dots, 1)$.

The DLB problem can be regarded as a deceptive version of the classic and well-studied LEADINGONES problem (Rudolph 1997; Droste, Jansen, and Wegener 2002; Böttcher, Doerr, and Neumann 2010), which counts the number of contiguous ones from left to right until the first zero is hit. Different from the unimodal LEADINGONES problem, the DLB problem has a huge number of local optima, however, with a very small basin of attraction.

We define a bi-objective DLB problem, called DLTB, in the same general way that Laumanns, Thiele, and Zitzler (2004) constructed the classic the bi-objective benchmark LOTZ from the LEADINGONES problems. The first objective function is the original DLB problem and the second objective function is the DLB problem with the roles of zeroes and ones exchanged and the order of the bits positions inverted.

Definition 1. *Let $n \in \mathbb{N}$ be even. The* DLTB *function $f = (f_1, f_2) : \{0,1\}^n \to \mathbb{R}^2$ is defined by*

$$f_1(x) = \begin{cases} 2m+1, & \text{if } x_{2m+1} + x_{2m+2} = 0, \\ 2m, & \text{if } x_{2m+1} + x_{2m+2} = 1, \\ n, & \text{if } x = 1^n, \end{cases}$$

$$f_2(x) = \begin{cases} 2k+1, & \text{if } x_{n-2k} + x_{n-(2k+1)} = 2, \\ 2k, & \text{if } x_{n-2k} + x_{n-(2k+1)} = 1, \\ n, & \text{if } x = 0^n, \end{cases}$$

for $x = (x_1, \dots, x_n)$ with $x_{2m+1}x_{2m+2}$ being the leftmost block whose value is not 11 (for $x \neq 1^n$) and $x_{n-(2k+1)}x_{n-2k}$ being the rightmost block whose value is not 00 (for $x \neq 0^n$). We call $x_{2m+1}x_{2m+2}$ and $x_{n-(2k+1)}x_{n-2k}$ critical blocks.

We determine the Pareto set and Pareto front of the DLTB problem. Due to the limited space, all proofs are omitted, but will be made available in an arXiv preprint.

Lemma 2. *The Pareto set of* DLTB *function is $S^* = \{1^{2a}0^{n-2a} \mid a \in [0..n/2]\}$. The Pareto front is $F^* = \{(2a+1, n-2a+1) \mid a \in [1..n/2-1]\} \cup \{(1,n),(n,1)\}$.*

Moreover, we calculate the number of different function values of the DLTB problem.

Lemma 3. *There are $\frac{1}{2}n^2 + 1$ different function values for the* DLTB *problem with problem size n.*

We recall that x is a *local optimum* of a single-objective problem $g : \{0,1\}^n \to \mathbb{R}$ if all Hamming neighbors of x have a smaller g-value that x. The single-objective g is *multimodal* if it has local optima different from the global optimum. Following (Doerr and Zheng 2021; Zheng and Doerr 2023c), a multi-objective function f is *multimodal* if at least one of its objective functions is multimodal.

Note that f_1 has the local optima 1^n and $(11)^i(00)*$ for $i \in [0..n/2-1]$ and $* \in \{0,1\}^{n-2i-2}$. Likewise, f_2 has the local optima 0^n and $*(11)(00)^i$ for $i \in [0..n/2-1]$ and $* \in \{0,1\}^{n-2i-2}$. Since only 1^n and 0^n, respectively, are global optima, the DLTB problem is multimodal.

We note that the above definition of multimodality only considers the modality of each objective. The following stronger definition uses domination to define local optima of multi-objective problems. We call x a *strong local optimum* of the multi-objective problem f if for any Hamming neighbor y of x we have $y \preceq x$. Now f is called *strongly multimodal* if it has strong local optima other than the Pareto optima. The following lemma shows that DLTB is strongly multimodal.

Lemma 4. *The set of strong local optima of the* DLTB *with problem size n is $L = \{1^{2a}00 * 110^{2b} \mid * \in \{0,1\}^{n-2a-2b-4}, a, b \in [0..n/2-2], 2a+2b+4 \leq n\} \cup S^*$.*

We further note that the DLTB problem admits pairwise non-dominating sets larger than the Pareto front.

Theorem 5. *Consider any set of solutions P such that $x \not\preceq y$ w.r.t.* DLTB *problem for all $x, y \in P$ with $x \neq y$. Then $|P| \leq n-1$, and this estimate is tight.*

In summary, the proposed DLTB has two features different from the established benchmarks in the theory of multi-objective EAs such as ONEMINMAX, COUNTINGONES-COUNTINGZEROES, LOTZ, or OJZJ. As a multi-objective multimodal function, it is the first one to contain (strong) local optima that are not already Pareto optima. As a multi-objective benchmark, DLTB is the first to have a Pareto front of size smaller than the largest size of a mutually non-dominating set of individuals.

In the following, we will analyze the expected number of function evaluations for the population to cover the full Pareto front for the first time. That is, denoting by P_t the population in the t-th generation, our aim is to analyze the *runtime* $T = \min\{t \geq 0 \mid F^* \subseteq f(P_t)\}$.

Algorithm 4: Metropolis algorithm with parent replacement and one-bit mutation to maximize function $f = (f_1, f_2)$

Generate x uniformly in $\{0,1\}^n$ and set $P = \{x\}$
loop
  Choose x uniformly at random from P and obtain x' via flipping one bit of x uniformly at random
  if there is no $y \in P$ such that $x' \prec y$ **then**
    $P = \{z \in P \mid z \not\preceq x'\} \cup \{x'\}$
  else
    Choose $b \in \{0,1\}$ randomly with $\Pr[b = 1] = \alpha^{f_1(x')-f_1(x)+f_2(x')-f_2(x)}$
    if $b = 1$ **then**
      $P = P \setminus \{x\} \cup \{x'\}$
    end if
  end if
end loop

Algorithm 5: Metropolis algorithm with parent replacement and standard bit-wise mutation to maximize $f = (f_1, f_2)$

Generate x uniformly in $\{0,1\}^n$ and set $P = \{x\}$
loop
  Choose x uniformly at random from P and obtain x' via flipping each bit value of x with probability $1/n$
  if there is no $y \in P$ such that $x' \prec y$ **then**
    $P = \{z \in P \mid z \not\preceq x'\} \cup \{x'\}$
  else
    Choose $b \in \{0,1\}$ randomly with $\Pr[b = 1] = \alpha^{f_1(x')-f_1(x)+f_2(x')-f_2(x)}$
    if $b = 1$ **then**
      $P = P \setminus \{x\} \cup \{x'\}$
    end if
  end if
end loop

Multi-Objective Metropolis Algorithm with Parent Replacement

The Metropolis algorithm shares many similarities with the RLS heuristic for single-objective optimization. Hence, the multi-objective Metropolis algorithm discussed in this paper will be built in a manner similar to the SEMO, the multi-objective counterpart of RLS. Naturally, this variant of the Metropolis algorithm will remove the parent when the offspring succeeds in entering the population. However, this section will show that this variant cannot efficiently cover the Pareto front of the DLTB, regardless of whether we use one-bit mutation or standard bit-wise mutation.

Inefficiency of One-Bit Mutation

As discussed above, we resort to the SEMO (Algorithm 3) as template to define a multi-objective variant of the Metropolis algorithm. Same as the SEMO, our multi-objective Metropolis algorithm starts with a randomly generated individual. In each iteration, an offspring is generated by applying one-bit mutation to a randomly picked solution from the current population. If this offspring is not dominated by any individual in the population, then it will enter the population and cause the removal of all individuals that are dominated by it. Inherited from the single-objective Metropolis algorithm, an inferior offspring still has the chance to enter the population in the multi-objective variant (different from SEMO). The probability for this event is determined by the function value difference between the offspring and its parent. In this first work, we simply sum up the function value differences in the objectives, but other kinds of acceptance probabilities could be interesting as well. We then follow the way of the single-objective Metropolis algorithm that an inferior offspring replaces its parent when it survives. See Algorithm 4 for the pseudocode of this algorithm.

The following theorem shows that the population size for Algorithm 4 is always one when optimizing DLTB, and thus cannot cover the full Pareto front.

Theorem 6. *Consider using Algorithm 4 to solve* DLTB. *Then the population size is always one, and thus the full Pareto front (with size $n/2 + 1$) will never be covered.*

Since any search trajectory of the SEMO with positive probability appears as search trajectory of the multi-objective Metropolis algorithm, Theorem 6 immediately implies the following result.

Theorem 7. *The SEMO algorithm cannot optimize the DLTB problem.*

Inefficiency of Standard Bit-Wise Mutation

The subsection above showed that with one-bit mutation, our Metropolis algorithm cannot solve the DLTB problem. One could speculate that the difficulties of the Metropolis algorithm are due to the locality of one-bit mutation, and that a global operator like standard bit-wise mutation will improve the situation. This subsection will give a mostly negative answer (even if our lower bound is less drastic than before).

So we shall now discuss the variant of Algorithm 4 where we replace one-bit mutation (Step 3 in Algorithm 4) by standard bit-wise mutation (flipping each bit independently with probability $1/n$) and keep other steps unchanged (Algorithm 5).

The following gives a calculation of the size of sets of non-dominated solutions that is more fine-grained than Theorem 5.

Theorem 8. *Consider any set of solutions P such that $x \not\preceq y$ w.r.t.* DLTB *problem for all $x, y \in P$ with $x \neq y$. Let $P^* = S^* \cap P$ be the set of Pareto optima in the population. Then $|P \setminus P^*| \leq n + 3 - 2|P^*|$ and $|P| \leq n + 3 - |P^*|$.*

Note that the population size increases only when an offspring is generated that is not dominated by the current population. Also, we recall that an accepted inferior offspring replaces its parent. This leads to the following estimate of the population size for Algorithm 5 when the number of covered Pareto optima is given. Note that this is not a trivial consequence of Theorem 8 since, due to the acceptance of inferior solutions, the population of the Metropolis algorithm may contain dominating solutions.

Corollary 9. *Consider using Algorithm 5 to solve the* DLTB *with problem size n. Let P be the current population*

and $P^* = S^* \cap P$ be the set of Pareto optima in the population. Then $|P \setminus P^*| \leq n+3-2|P^*|$ and $|P| \leq n+3-|P^*|$.

From these structural insights, we can show the following runtime results.

Lemma 10. *Let n be sufficiently large and the the acceptance parameter be $\alpha = e$. Consider using Algorithm 5 to solve the* DLTB. *Assume that the current population P covers $0.495(n+1)$ Pareto optima. Then it needs $\Omega(n^5)$ iterations in expectation to cover the full Pareto front.*

Note that in each iteration the number of the covered Pareto front points can change by at most 1 as only one individual is generated in each iteration. Hence, the number of $0.495(n+1)$ covered Pareto optima must be reached at some time before the full coverage. Thus, from Lemma 10, we obtain the following theorem.

Theorem 11. *Let n be sufficiently large and $\alpha = e$. Then Algorithm 5 needs $\Omega(n^5)$ time in expectation to cover the full Pareto front for* DLTB.

DLTB Is Not Hard to Solve

The previous section shows that when the inferior offspring replaces its parent in case of acceptance, the multi-objective Metropolis algorithm cannot efficiently cover the full Pareto front of DLTB, both with the local mutation operator of one-bit mutation and the global operator of standard bit-wise mutation. Noting the distinctive properties of the DLTB problem discussed before, one may suspect that it is simply the difficulty of the problem that causes the poor performance.

In this section, we refute this speculation and show that the GSEMO, NSGA-II, and SMS-EMOA all perform reasonably well on DLTB.

An $O(n^4)$ Runtime Guarantee for the GSEMO

The GSEMO algorithm (Giel 2003) is similar to the SEMO, the only difference being that is utilizes the standard bit-wise mutation (flipping each bit independently with probability $1/n$) instead of one-bit mutation. We note that the only difference to the multi-objective Metropolis variant Algorithm 5 is that the GSEMO will not let any dominated offspring into the population (without Steps 7-10 in Algorithm 5).

The optimization process of GSEMO optimizing the DLTB can be divided into two phases. The first phase is to reach a Pareto front point for the first time, and the second one is to cover the whole Pareto front. Lemma 12 pessimistically considers the time to reach 1^n as the first reached Pareto optimum, and shows an upper bound of $O(n^4)$ iterations in expectation to reach this goal.

Lemma 12. *Consider using the GSEMO algorithm to optimize* DLTB *problem with problem size n. Then within expected iterations of $en^3(n-1)$ at least one Pareto front point will be reached.*

Note that already reached Pareto front points will be maintained in all future generations. Lemma 13 proves that within $\frac{en^3(n-1)}{2}$ iterations in expectation, the full Pareto front will be covered once one Pareto front point is reached.

Algorithm 6: NSGA-II to maximize $f : \{0,1\}^n \to \mathbb{R}^m$

1: Uniformly at random generate the initial population $P_0 = \{x_1, x_2, \ldots, x_N\}$ with $x_i \in \{0,1\}^n, i = 1, 2, \ldots, N$
2: **for** $t = 0, 1, 2, \ldots$ **do**
3: Generate the offspring population Q_t with size N
4: Use fast-non-dominated-sort() (Deb et al. 2002) to divide $R_t = P_t \cup Q_t$ into $F_1, F_2, \ldots$
5: Find $i^* \geq 1$ such that $\sum_{i=1}^{i^*-1} |F_i| < N$ and $\sum_{i=1}^{i^*} |F_i| \geq N$
6: Separately calculate the crowding distance of each individual in $F_1, \ldots, F_{i^*}$
7: Let $\tilde{F}_{i^*}$ be the $N - \sum_{i=1}^{i^*-1} |F_i|$ individuals in F_{i^*} with largest crowding distance, chosen at random in case of a tie
8: $P_{t+1} = (\bigcup_{i=1}^{i^*-1} F_i) \cup \tilde{F}_{i^*}$
9: **end for**

Lemma 13. *Consider using the GSEMO algorithm to optimize the* DLTB *problem with problem size n. Assume that a Pareto optimum has been found. Then the expected time to cover the full Pareto front is at most $\frac{en^3(n-1)}{2}$.*

Hence, we have the following corollary for the runtime of GSEMO on DLTB from Lemmas 12 and 13.

Corollary 14. *The expected runtime of the GSEMO algorithm on the* DLTB *benchmark with problem size n is at most $\frac{3en^3(n-1)}{2}$.*

An $O(n^4)$ Runtime Guarantee for the NSGA-II

The NSGA-II (Deb et al. 2002) is the most applied multi-objective evolutionary algorithm with more than 50,000 citations on Google scholar. The first runtime analysis were obtained in (Zheng, Liu, and Doerr 2022; Zheng and Doerr 2023a), quickly followed up by an impressive set of works including (Bian and Qian 2022; Doerr and Qu 2023; Dang et al. 2023; Zheng and Doerr 2023b). Different from the discussed Metropolis algorithms and (G)SEMO with the possibility of changing population size, NSGA-II uses a fixed population size N. Starting from a parent population P_t, the offspring population Q_t with the same size of N is generated. For the combined population $R_t = P_t \cup Q_t$, N individuals must be removed for the next population. It divided R into several fronts $F_1, F_2, \ldots$. All individuals in smaller fronts than critical front F_{i^*} will be kept to P_{t+1}, and only $N - \sum_{i=1}^{i^*-1} |F_i|$ individuals with largest crowding distance values in F_{i^*} will survive to P_t. See Algorithm 6.

Note that already-reached Pareto front points are possible to be removed when a population size is not chosen properly. The following lemma shows that when the population size is at least $4(n-1)$, any function value covered by the first front (also including the covered Pareto front points) will be maintained once it is reached. It stems from the fact that any function value has at most 4 corresponding individuals with positive crowding distance values.

Algorithm 7: SMS-EMOA to maximize $f : \{0,1\}^n \to \mathbb{R}^m$

1: Generate P_0 by selecting μ solutions uniformly and randomly from $\{0,1\}^n$ with replacement
2: **for** $t = 0, 1, 2, \ldots,$ **do**
3: Select a solution x uniformly at random from P_t
4: Generate x' by flipping each bit of x independently with probability $1/n$
5: Use fast-non-dominated-sort() (Deb et al. 2002) to divide $R_t = P_t \cup \{x'\}$ into $F_1, \ldots, F_{i^*}$
6: Calculate $\Delta_r(z, F_{i^*})$ for all $z \in F_{i^*}$ and find $D = \arg\min_{z \in F_{i^*}} \Delta_r(z, F_{i^*})$
7: Uniformly at random pick $z' \in D$ and $P_{t+1} = R_t \setminus \{z'\}$
8: **end for**

Lemma 15. *Consider one iteration of the NSGA-II with population size $N \geq 4(n-1)$ optimizing the* DLTB. *Assume that in some iteration t the combined parent and offspring population $R_t = P_t \cup Q_t$ contains a solution x with rank one. Then also the next parent population P_{t+1} contains an individual y with $f(y) = f(x)$. In particular, once the parent population contains an individual z with objective value $f(z) \in F^*$, it will do so for all future generations.*

Then we have the following theorem.

Theorem 16. *Consider optimizing* DLTB *function via the NSGA-II with one of the following two ways to generate the offspring population in Step 3 of Algorithm 6, namely applying standard bit-wise mutation once to each parent or N times choosing a parent uniformly at random and standard bit-wise mutation to it. If the population size N is at least $4(n-1)$, then the expected runtime is $O(n^3)$ iterations and $O(Nn^3)$ fitness evaluations.*

An $O(n^4)$ Runtime Guarantee for the SMS-EMOA

The SMS-EMOA algorithm (Beume, Naujoks, and Emmerich 2007) is a variant of the steady-state NSGA-II that replaces the crowding distance by the hypervolume measure. The runtime for multi-objective optimization is first analyzed in (Bian et al. 2023), and (Zheng and Doerr 2024) further analyzes how it performs for many objectives. Same as the NSGA-II, SMS-EMOA uses a fixed population size μ. Different from NSGA-II, each time SMS-EMOA generates one offspring, similar to the multi-objective Metropolis algorithms and (G)SEMO. One individual in the combined population R_t will be removed in the survival selection. With the same partition of R_t into $F_1, \ldots, F_{i^*}$, the individual in F_{i^*} with the smallest hypervolume contribution is removed. Formally, let $D = \arg\min_{z \in F_{i^*}} \Delta_r(z, F_{i^*})$ where $\Delta_r(z, F_{i^*}) = \mathrm{HV}_r(F_{i^*}) - \mathrm{HV}_r(F_{i^*} \setminus \{z\})$ and HV_r is the hypervolume value with reference point r and for $\mathbb{R}^2$ it is calculated as $\mathrm{HV}_r(S) = \mathcal{L}\left(\bigcup_{u \in S} H_{u,r}\right)$, where $\mathcal{L}$ is the Lebesgue measure and $H_{u,r} = [r_1, u_1] \times [r_2, u_2]$ is the rectangle defined by the corners u and r. The one individual in D is uniformly at random picked to be removed. See Algorithm 7 for more details.

The following lemma shows that when the population size is at least $n-1$, any function value covered by the first front (also including the covered Pareto front points) will be maintained once it is reached.

Algorithm 8: Metropolis algorithm with keeping the parent to maximize $f = (f_1, f_2)$

1: Generate a search point x uniformly in $\{0,1\}^n$ and $P = \{x\}$
2: **loop**
3: Choose x uniformly at random from P and obtain x' via one-bit mutation or bit-wise mutation
4: **if** there is no $y \in P$ such that $x' \prec y$ **then**
5: $P = \{z \in P \mid z \not\preceq x'\} \cup \{x'\}$
6: **else**
7: Choose $b \in \{0,1\}$ randomly with $\Pr[b = 1] = \alpha^{f_1(x') - f_1(x) + f_2(x') - f_2(x)}$
8: **if** $b = 1$ **then**
9: $P = \{z \in P \mid z \not\preceq x'\} \cup \{x'\}$
10: **end if**
11: **end if**
12: **end loop**

Lemma 17. *Consider using SMS-EMOA with population size $\mu \geq n-1$ to optimize* DLTB. *Then any function value covered by F_1 (Step 5 in Algorithm 7) will survive to the next generation. In particular, any Pareto front point (also any Pareto optimum) will always be covered once it is reached.*

Theorem 18. *Consider using SMS-EMOA with population size $\mu \geq n-1$ to optimize* DLTB. *Then the Pareto front will be covered in at most $3e\mu n^3/2$ iterations in expectation.*

Multi-Objective Metropolis Algorithm with Keeping the Parent

Given the difficulty of the previous Metropolis variant and the efficiency of other algorithms in previous sections this section will discuss whether there is a Metropolis variant efficiently solves the multi-objective optimization. We give a positive answer for a variant with keeping the parent.

We resort to the multi-objective simulated annealing (MOSA) (the category that the Metropolis algorithm belongs to) in application works, expecting an efficient Metropolis variant. The MOSAs (Suman and Kumar 2006; Bandyopadhyay et al. 2008) usually maintain an archive to store the non-dominated solutions. However, for the Metropolis variants discussed before the parent (even if it cannot be dominated by others) will be removed when the inferior offspring survives. Hence, we consider the variant of Algorithms 4 and 5 with the modification that when the inferior offspring survives, only the individuals that are weakly dominated by it will be removed, see Step 9 in Algorithm 8. Note that MOSAs usually contain additional features, like considering the number of dominating solutions. This paper will not analyze the Metropolis variants with such features, since we try to focus on the essential components, but will set them as our future work.

It is not difficult to see that for Algorithm 8, any two solutions with the same function value will not co-exist in the population due to the survival selection (Step 5 or 9). From

Lemma 3 that there are $\frac{1}{2}n^2 + 1$ different function values, we know that the population size will be at most $\frac{1}{2}n^2 + 1$. We formulate it in the following lemma.

Lemma 19. *Consider using Algorithm 8 to optimize DLTB. Then* $|P| \leq \frac{1}{2}n^2 + 1$.

Note that for Algorithm 8, in the combined population of the current population and the offspring, any non-dominated solution will not be removed in the survival selection. Hence, we pessimistically resort to the proof ideas in deriving the runtime of GSEMO (Lemmas 12 and 13), and easily obtain the following lemma via replacing the upper bound of $|P|$ from $n-1$ used there to $\frac{1}{2}n^2+1$ as discussed.

Theorem 20. *Consider using Algorithm 8 with standard bit-wise mutation to optimize the* DLTB *problem with problem size* n. *Then the expected time is at most* $\frac{3e}{4}n^5 + \frac{3e}{2}n^3$.

We note that the trivial runtime upper bound of $O(n^5)$ is not tight. The upper bound of the population size stems from the fact that it is possible to generate a population containing at most $n - v_1 + 1$ different individuals (f_2 values) with the same f_1 value of v_1 (like generating the f_2 value from the largest to the smallest, which are all saved due to Step 9 in Algorithm 8) from Lemma 3. However, for a given f_1 value, if a solution with a larger f_2 value is generated, all solutions with smaller f_2 values will be removed due to Step 5 or 9. Hence, we conjecture that the population size will not be as large as $\Theta(n^2)$ with high probability.

For the one-bit mutation, we conjecture a better result as it will result in a good chance of the survival of the bridge point between two neighbor Pareto optima (e.g. 11010000 is a bridge point between 11000000 and 11110000). Consider the case when a Pareto optimum x is reached but its neighbor Pareto optimum y is not reached. If the bridge point is selected for mutation, then the probability of reaching y is $1/n$, instead of the probability of $\Theta(1/n^2)$ reaching y from x with standard bit-wise mutation. Unfortunately, we do not have formal proof for this and the above population size argument. We will consider them in our future work and in this paper conduct some experiments to test the actual runtime.

Experiments

The previous sections show the inefficiency of the multi-objective Metropolis with parent replacement, the efficiency of the GSEMO, NSGA-II, and SMS-EMOA, and the efficiency of the multi-objective Metropolis keeping the parent. In this section, we experimentally verify them. For the variants with parent replacement, the one using one-bit mutation (Algorithm 4) will surely have a population size of one, and we will not report its runtime. Due to the inefficiency of such a variant with the standard bit-wise mutation (Algorithm 5), we run Algorithm 5 with a maximal number of function evaluations of 10^8 in 20 independent runs. Other algorithms were conducted with 50 independent runs and terminated when the full Pareto front was covered. We set $\alpha = 3$ as in the paper (Wang, Zheng, and Doerr 2024) of the single-objective Metropolis algorithm, set $N = 4n$ (at least $4(n-1)$ as suggested in Theorem 16), fair selection, and standard bit-wise mutation for NSGA-II, and set $\mu = n - 1$ (at least $n - 1$ as suggested in Theorem 18).

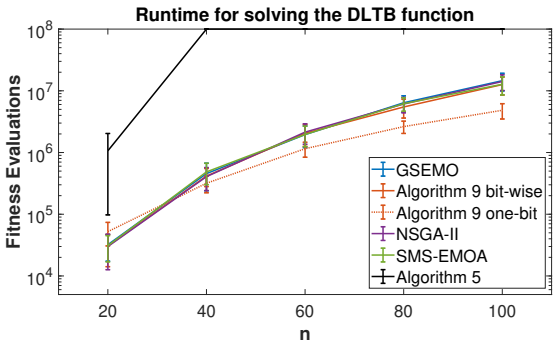

Figure 1: The mean (with standard deviations) number of fitness evaluations of GSEMO, Metropolis variants (Algorithm 5, Algorithm 8 with bit-wise mutation, Algorithm 8 with one-bit mutation), NSGA-II, and SMS-EMOA for solving DLTB with $n \in \{20, 40, 60, 80, 100\}$ in 50 independent runs (20 runs for Algorithm 5).

Figure 1 shows the runtime of these algorithms. We easily see the inefficiency of the Metropolis variant with parent replacement (Algorithm 5) and the efficiency and similar runtime of the GSEMO, NSGA-II, and SMS-EMOA. We also note that the Metropolis variant keeping the parent with the standard bit-wise mutation (orange solid line) has a similar runtime to GSEMO, which indicates the possibility of improving the upper bound in Theorem 20 in the future. More interestingly, this variant with the one-bit mutation (orange dashed line) shows better performance, which verifies the discussions at the end of the previous section and encourages us for future theoretical analysis.

Conclusion

This paper discussed how to use the Metropolis algorithm for multi-objective optimization. A bi-objective DLB benchmark was proposed, which is the first multi-objective multimodal benchmark that contains local optima which are not Pareto optima. Also, it is the first multi-objective benchmark having a Pareto front smaller than the size of a set of mutually non-dominated objective values.

We used this benchmark to design effective multi-objective Metropolis algorithms. We observed that no good performance could be obtained when inferior accepted offspring replace the parent (as in the single-objective Metropolis algorithm). However, we also showed that keeping the parent in this situation gives efficient algorithms for the bi-objective DLB benchmark. As a side results, we proved runtime guarantees for the GSEMO, NSGA-II, and SMS-EMOA on this new benchmark.

Our experiments support the above findings and suggest that the second multi-objective Metropolis algorithm is more efficient than what our proven guarantees suggest.

Acknowledgments

This work was supported by National Natural Science Foundation of China (Grant No. 62306086), Science, Technology and Innovation Commission of Shenzhen Municipality (Grant No. GXWD20220818191018001), Guangdong Basic and Applied Basic Research Foundation (Grant No. 2019A1515110177). This research benefited from the support of the FMJH Program Gaspard Monge for optimization and operations research and their interactions with data science.

References

Antipov, D.; and Doerr, B. 2021. A tight runtime analysis for the $(\mu + \lambda)$ EA. *Algorithmica*, 83: 1054–1095.

Auger, A.; and Doerr, B., eds. 2011. *Theory of Randomized Search Heuristics*. World Scientific Publishing.

Bandyopadhyay, S.; Saha, S.; Ujjwal, M.; and Deb, K. 2008. A simulated annealing-based multiobjective optimization algorithm: AMOSA. *IEEE Transactions on Evolutionary Computation*, 12: 269–283.

Beume, N.; Naujoks, B.; and Emmerich, M. 2007. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181: 1653–1669.

Bian, C.; and Qian, C. 2022. Better running time of the non-dominated sorting genetic algorithm II (NSGA-II) by using stochastic tournament selection. In *Parallel Problem Solving From Nature, PPSN 2022*, 428–441. Springer.

Bian, C.; Zhou, Y.; Li, M.; and Qian, C. 2023. Stochastic population update can provably be helpful in multi-objective evolutionary algorithms. In *International Joint Conference on Artificial Intelligence, IJCAI 2023*, 5513–5521. ijcai.org.

Böttcher, S.; Doerr, B.; and Neumann, F. 2010. Optimal fixed and adaptive mutation rates for the LeadingOnes problem. In *Parallel Problem Solving from Nature, PPSN 2010*, 1–10. Springer.

Dang, D.-C.; Opris, A.; Salehi, B.; and Sudholt, D. 2023. A proof that using crossover can guarantee exponential speedups in evolutionary multi-objective optimisation. In *Conference on Artificial Intelligence, AAAI 2023*, 12390–12398. AAAI Press.

Deb, K.; Pratap, A.; Agarwal, S.; and Meyarivan, T. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6: 182–197.

Doerr, B.; El Ghazi El Houssaini, T.; Rajabi, A.; and Witt, C. 2023. How well does the Metropolis algorithm cope with local optima? In *Genetic and Evolutionary Computation Conference, GECCO 2023*, 1000–1008. ACM.

Doerr, B.; and Krejca, M. S. 2020. Significance-based estimation-of-distribution algorithms. *IEEE Transactions on Evolutionary Computation*, 24: 1025–1034.

Doerr, B.; and Krejca, M. S. 2021. The univariate marginal distribution algorithm copes well with deception and epistasis. *Evolutionary Computation*, 29: 543–563.

Doerr, B.; and Neumann, F., eds. 2020. *Theory of Evolutionary Computation—Recent Developments in Discrete Optimization*. Springer. Also available at http://www.lix.polytechnique.fr/Labo/Benjamin.Doerr/doerr_neumann_book.html.

Doerr, B.; and Qu, Z. 2023. From understanding the population dynamics of the NSGA-II to the first proven lower bounds. In *Conference on Artificial Intelligence, AAAI 2023*, 12408–12416. AAAI Press.

Doerr, B.; and Zheng, W. 2021. Theoretical analyses of multi-objective evolutionary algorithms on multi-modal objectives. In *Conference on Artificial Intelligence, AAAI 2021*, 12293–12301. AAAI Press.

Droste, S.; Jansen, T.; and Wegener, I. 2000. Dynamic parameter control in simple evolutionary algorithms. In *Foundations of Genetic Algorithms, FOGA 2000*, 275–294. Morgan Kaufmann.

Droste, S.; Jansen, T.; and Wegener, I. 2002. On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*, 276: 51–81.

Giel, O. 2003. Expected runtimes of a simple multi-objective evolutionary algorithm. In *Congress on Evolutionary Computation, CEC 2003*, 1918–1925. IEEE.

Jansen, T. 2013. *Analyzing Evolutionary Algorithms – The Computer Science Perspective*. Springer.

Jansen, T.; Jong, K. A. D.; and Wegener, I. 2005. On the choice of the offspring population size in evolutionary algorithms. *Evolutionary Computation*, 13: 413–440.

Jansen, T.; and Wegener, I. 2007. A comparison of simulated annealing with a simple evolutionary algorithm on pseudo-Boolean functions of unitation. *Theoretical Computer Science*, 386: 73–93.

Laumanns, M.; Thiele, L.; and Zitzler, E. 2004. Running time analysis of multiobjective evolutionary algorithms on pseudo-Boolean functions. *IEEE Transactions on Evolutionary Computation*, 8: 170–182.

Lehre, P. K.; and Nguyen, P. T. H. 2019. On the limitations of the univariate marginal distribution algorithm to deception and where bivariate EDAs might help. In *Foundations of Genetic Algorithms, FOGA 2019*, 154–168. ACM.

Lissovoi, A.; Oliveto, P. S.; and Warwicker, J. A. 2023. When move acceptance selection hyper-heuristics outperform Metropolis and elitist evolutionary algorithms and when not. *Artificial Intelligence*, 314: 103804.

Metropolis, N.; Rosenbluth, A. W.; Rosenbluth, M. N.; Teller, A. H.; and Teller, E. 1953. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21: 1087–1092.

Neumann, F.; and Witt, C. 2010. *Bioinspired Computation in Combinatorial Optimization – Algorithms and Their Computational Complexity*. Springer.

Oliveto, P. S.; Paixão, T.; Heredia, J. P.; Sudholt, D.; and Trubenová, B. 2018. How to escape local optima in black box optimisation: when non-elitism outperforms elitism. *Algorithmica*, 80: 1604–1633.

Rudolph, G. 1997. *Convergence Properties of Evolutionary Algorithms*. Verlag Dr. Kovăc.

Sasaki, G. H.; and Hajek, B. 1988. The time complexity of maximum matching by simulated annealing. *Journal of the ACM*, 35: 387–403.

Sudholt, D.; and Witt, C. 2019. On the choice of the update strength in estimation-of-distribution algorithms and ant colony optimization. *Algorithmica*, 81: 1450–1489.

Suman, B.; and Kumar, P. 2006. A survey of simulated annealing as a tool for single and multiobjective optimization. *Journal of the Operational Research Society*, 57: 1143–1160.

Wang, S.; Zheng, W.; and Doerr, B. 2024. Choosing the right algorithm with hints from complexity theory. *Information and Computation*, 296: 105125.

Witt, C. 2006. Runtime analysis of the $(\mu + 1)$ EA on simple pseudo-Boolean functions. *Evolutionary Computation*, 14: 65–86.

Zheng, W.; and Doerr, B. 2023a. Mathematical runtime analysis for the non-dominated sorting genetic algorithm II (NSGA-II). *Artificial Intelligence*, 325: 104016.

Zheng, W.; and Doerr, B. 2023b. Runtime analysis for the NSGA-II: proving, quantifying, and explaining the inefficiency for many objectives. *IEEE Transactions on Evolutionary Computation*. In press, https://doi.org/10.1109/TEVC.2023.3320278.

Zheng, W.; and Doerr, B. 2023c. Theoretical analyses of multiobjective evolutionary algorithms on multimodal objectives. *Evolutionary Computation*, 31: 337–373.

Zheng, W.; and Doerr, B. 2024. Runtime analysis of the SMS-EMOA for many-objective optimization. In *Conference on Artificial Intelligence, AAAI 2024*. AAAI Press.

Zheng, W.; Liu, Y.; and Doerr, B. 2022. A first mathematical runtime analysis of the Non-Dominated Sorting Genetic Algorithm II (NSGA-II). In *Conference on Artificial Intelligence, AAAI 2022*, 10408–10416. AAAI Press.

Zhou, Z.-H.; Yu, Y.; and Qian, C. 2019. *Evolutionary Learning: Advances in Theories and Algorithms*. Springer.