

Speeding up Local Search for the Indicator-based Subset Selection Problem by a Candidate List Strategy

Keisuke Korogi, and Ryoji Tanabe, *Member, IEEE*

Abstract—In evolutionary multi-objective optimization, the indicator-based subset selection problem involves finding a subset of points that maximizes a given quality indicator. Local search is an effective approach for obtaining a high-quality subset in this problem. However, local search requires high computational cost, especially as the size of the point set and the number of objectives increase. To address this issue, this paper proposes a candidate list strategy for local search in the indicator-based subset selection problem. In the proposed strategy, each point in a given point set has a candidate list. During search, each point is only eligible to swap with unselected points in its associated candidate list. This restriction drastically reduces the number of swaps at each iteration of local search. We consider two types of candidate lists: nearest neighbor and random neighbor lists. This paper investigates the effectiveness of the proposed candidate list strategy on various Pareto fronts. The results show that the proposed strategy with the nearest neighbor list can significantly speed up local search on continuous Pareto fronts without significantly compromising the subset quality. The results also show that the sequential use of the two lists can address the discontinuity of Pareto fronts.

Index Terms—Evolutionary multi-objective optimization, indicator-based subset selection, local search

I. INTRODUCTION

MULTI-OBJECTIVE optimization aims to simultaneously minimize d objective functions f_1, \dots, f_d . In general, no absolute optimal solution can minimize all d objective functions. Thus, the ultimate goal of multi-objective optimization is to find a Pareto optimal solution preferred by a decision maker [1]. An a posteriori decision making is generally performed when the decision maker's preference information is not available, where she/he selects a single solution from a non-dominated solution set. Evolutionary multi-objective optimization (EMO) [2] is an effective approach for obtaining a non-dominated solution set that approximates the Pareto front (PF) in the objective space. For simplicity, this paper denotes a d -dimensional objective vector $f(x)$ of a solution x as a point p . This paper also considers only the objective space $V \subseteq \mathbb{R}^d$.

Given a non-dominated point set $P \subseteq V$ of size n , a quality indicator \mathcal{I} , and a positive integer k , the indicator-based subset selection problem (ISSP) [3] aims to find a subset $S^* \subset P$

K. Korogi is with Graduate School of Environment and Information Sciences, Yokohama National University, Yokohama, Japan. (e-mail: keisuke.korogi.52@gmail.com).

R. Tanabe is with Faculty of Environment and Information Sciences, Yokohama National University, Yokohama, Japan. (e-mail: rt.ryoji.tanabe@gmail.com).

of size k that maximizes \mathcal{I} , where $k < n$. As discussed in [3], [4], the ISSP can be generally found in the context of EMO. For example, previous studies [5], [6], [7] reported the effectiveness of an unbounded external archive that maintains all non-dominated solutions found so far. Although the size of the unbounded external archive is generally large at the end of the search, it is difficult for the decision maker to examine such a large number of solutions. Thus, it is necessary to select a small number of representative solutions from the archive for her/him. This postprocessing procedure is identical to the ISSP. In addition, the ISSP appears in environmental selection in indicator-based EMO algorithms [4]. Let P and Q be the population and offspring, respectively. Let also k and n be the sizes of P and the union $P \cup Q$, respectively. In environmental selection in indicator-based EMO algorithms (e.g., HypE [8]),¹ the best k individuals in terms of a quality indicator are needed to be selected from n individuals in $P \cup Q$. This is exactly the same as the ISSP. Any subset selection method for the ISSP can be incorporated into indicator-based EMO algorithms in a plug-in manner. Thus, designing an effective subset selection method is beneficial from the perspective of indicator-based EMO algorithms.

Representative inexact approaches for the ISSP include genetic algorithm [7], [12], local search (LS) [3], [13], [14], [15], and greedy search [3], [16], [17], [18]. The computation of a quality indicator \mathcal{I} is generally expensive as the number of objectives d and the subset size k increase. For this reason, the computationally cheap greedy search approach is the most popular in the ISSP. In fact, Basseur et al. [3] investigated the performance of LS but focused only on small-size ISSP instances with $n \leq 1000$. Bradstreet et al. [13] reported that LS performs poorly for a large k due to its high computational cost. Nan et al. [14] terminated the LS procedure early at 1 000 iterations to avoid a time-consuming process. In [15], they also terminated the LS procedure when the hypervolume improvement falls below a predetermined threshold. In both cases, there is no guarantee that LS finds a local optimal subset.

However, as demonstrated in [3], [13], LS for the ISSP requires high computational cost but can find a better subset than greedy search. Thus, an effective inexact method for the ISSP can be designed by reducing the computational cost in

¹Environmental selection in decomposition-based EMO algorithms [9] is based on scalarizing functions and generally does not use any quality indicator. Thus, the ISSP does not appear in decomposition-based EMO algorithms, and their selection methods (e.g., [10], [11]) cannot be applied to the ISSP.

LS. Roughly speaking, two approaches can be considered to speed up LS. One is to improve a method for evaluating a solution. Iterative stochastic search methods (including LS) require many solution evaluations. If the evaluation of a solution is time-consuming, it incurs a high computational cost. For example, the computation of the hypervolume indicator [19] is expensive as the number of objectives d increases. To address this issue, Bradstreet et al. [20] proposed a fast method for computing the hypervolume of a subset S in LS. Shang et al. [17] also proposed the use of the R2 indicator [21], [22] instead of the hypervolume indicator.² Here, the R2 indicator is computationally cheaper than the hypervolume indicator.

The other is to reduce the neighborhood size in LS. In other words, this approach reduces the number of candidate solutions to be evaluated by an objective function for each iteration. A candidate list strategy [23] is one of the most classical reduction approaches in the context of combinatorial optimization. As an example, let us consider LS with 2-opt moves in the travelling salesperson problem (TSP). Note that we below describe the candidate list strategy for the TSP just to explain the working concept of the candidate list strategy. Note also that it is impossible to re-use the candidate list strategy for the TSP on the ISSP in a straightforward manner. In the TSP, the simplest candidate list for each city includes its l nearest cities in terms of the Euclidean distance, where l is typically 10–40. Although LS scans the 2-opt neighborhood of the current tour for each iteration, the neighborhood size becomes significantly large as the number of cities increases. Here, it has been empirically observed that exchanging an edge connecting two nodes far from each other is unlikely to improve the tour length [23]. For this reason, even if LS restricts the neighborhood to candidate lists, it does not significantly affect the quality of tours.

While the first approach to speed up LS has been studied for the ISSP as described above, the second approach reducing the neighborhood size has attracted less attention in the context of the ISSP. LS for the ISSP could potentially be sped up by not examining “unpromising subsets” that do not lead to improvements. However, how to detect such “unpromising subsets” is not straightforward in the ISSP. A subset (i.e., a solution in the ISSP) is generally represented by a 0-1 vector. Each position in the binary vector indicates whether the corresponding point is selected. Unfortunately, this binary vector itself does not provide a clue for restricting the neighborhood.

Motivated by the above discussion, first, this paper analyzes the property of the ISSP to demonstrate the rationale for the neighborhood restriction. Then, based on the analysis, this paper proposes a candidate list strategy to speed up LS in the ISSP. Let l be a size of the candidate list. For each point p in the current subset S , the proposed candidate list strategy swaps only l unselected points in the candidate list of p . First-improvement LS with the 2-swap operator needs to examine $k(n - k)$ subsets at each iteration in the worst case [3]. In contrast, the proposed candidate list strategy can reduce this to kl subsets. Thus, the proposed candidate

²Strictly speaking, they addressed the ISSP using the R2 indicator, not the ISSP using the hypervolume indicator.

list strategy can speed up LS by saving unpromising swaps. Compared to the above-mentioned existing methods in the first approach, the advantage of the proposed candidate list strategy is generality with respect to quality indicators. For example, the methods [17], [20] can be applied to only the ISSP using the hypervolume indicator. In contrast, the proposed candidate list strategy can be applied to the ISSP using any quality indicator.

We consider two types of candidate lists: nearest neighbor and random neighbor lists. The concept of the nearest neighbor list is the same as that in the TSP. For each point p , its nearest neighbor list includes l nearest points in the objective space. In contrast, the random list includes l randomly selected points. We point out that LS with the nearest neighbor list does not work well on discontinuous PFs, e.g., the PF of DTLZ7 [24]. To address this issue, we propose a two-phase candidate list strategy that sequentially uses the random and nearest neighbor lists. We investigate the effectiveness of the proposed candidate list strategy on the ISSP of seven quality indicators, including the hypervolume [19] indicator.

The rest of this paper is organized as follows. Section II provides some preliminaries. Section III analyzes the property of the ISSP, where we focus on the distance between points for a successful swap. Section IV proposes the candidate list strategy. Section V describes the experimental setup. Section VI shows analysis results. Section VII concludes this paper.

II. PRELIMINARIES

A. Multi-objective optimization

Here, we consider the minimization of d objective functions $f = (f_1, \dots, f_d)$. $V \subseteq \mathbb{R}^d$ represents a d -dimensional objective space. As described in Section I, we denote an objective vector $f(x)$ as a point p , i.e., $p = f(x)$.

A point p is said to dominate another point q if $p_i \leq q_i$ for all $i \in \{1, \dots, d\}$ and $p_i < q_i$ for at least one index i . We denote this Pareto dominance relation as $p \prec q$. In addition, p is said to weakly dominate q if $p_i \leq q_i$ for all $i \in \{1, \dots, d\}$. If $p^* \in V$ is not dominated by any point in V , p^* is called a Pareto optimal point. The set of all Pareto optimal points $\{p^* \in V \mid \nexists p \in V : p \prec p^*\}$ is called the PF.

B. Quality indicators

Let $P \subset V$ be a non-dominated point set of size n found by an EMO algorithm. It is desirable that P approximates the PF well. Let Ω be the set of all non-dominated point sets in the objective space V . A quality indicator $\mathcal{I} : \Omega \rightarrow \mathbb{R}$ evaluates the quality of P in terms of at least one of the convergence, uniformity, and spread [25], [26], [27]. Here, a combination of the uniformity and the spread is called “diversity” in the EMO community. Since we focus only on the distribution of non-dominated points throughout this paper, we do not consider the cardinality. This paper also considers only unary quality indicators.

Representative quality indicators include hypervolume (HV) [19], inverted generational distance (IGD) [28], IGD plus (IGD^+) [29], the additive ϵ -indicator (ϵ) [26], R2 [21], new R2 indicator (NR2) [30], and s -energy [31]. For their details,

see [27], [32]. Briefly speaking, HV calculates the volume of the region dominated by the points in P and bounded by the reference point $r \in V$. IGD measures the average distance from each reference point $s \in S \subset V$ to its nearest point in P , where S is a set of reference points uniformly distributed on the PF. Since IGD is not Pareto-compliant, IGD can mislead the results [33], [34]. IGD^+ is a weakly Pareto-compliant version of IGD that uses a modified distance function. The ϵ indicator measures the minimum shift such that each point in P weakly dominates at least one reference point in S . R2 calculates the average minimum values of the weighted Tchebycheff function values of P with respect to a weight vector set W . NR2 is an improved version of R2 that more closely approximates the HV value of P . The s -energy indicator was originally proposed in the literature of mathematics. The s -energy indicator evaluates the uniformity of the points in P .

Let us consider the ranking of all point sets in V by a quality indicator \mathcal{I} . If the ranking by \mathcal{I} is consistent with the Pareto dominance relation, \mathcal{I} is said to be Pareto-compliant [35]. Since HV is one of the most popular Pareto-compliant indicators, HV is a reasonable first choice. However, HV does not accurately evaluate the uniformity of a point set in most cases [32], [36], [37]. Other quality indicators are generally used in combination to complement the results of HV.

As reviewed in [38], some preference-based quality indicators (e.g., R-HV and R-IGD [39]) have been proposed for benchmarking preference-based EMO algorithms [40]. Preference-based quality indicators take the decision maker's preference information into account in quality assessment. Thus, they differ from the general quality indicators (e.g., HV and IGD), which do not consider a priori preference information from the decision maker. Addressing preference-based quality indicators is beyond the scope of this paper.

C. Indicator-based subset selection problem

Given a d -dimensional objective space $V \subseteq \mathbb{R}^d$, the ISSP is to find a subset S with the maximum quality indicator value $\mathcal{I}(S)$:

$$S^* = \underset{S \subset P, |S|=k}{\operatorname{argmax}} \mathcal{I}(S), \quad (1)$$

where P is a set of n non-dominated points, and k is the size of S . Note that $k < n$. The quality indicator \mathcal{I} to be minimized (e.g., IGD and R2) can be reformulated as $-\mathcal{I}$ without loss of generality. In equation (1), the number of all possible subsets is $\binom{n}{k}$. Technically, a subset S can be represented by an n -dimensional 0-1 vector $u = (u_1, \dots, u_n)^\top$, where $\sum_{i=1}^n u_i = k$. For $i \in \{1, \dots, n\}$, if $u_i = 1$, S includes the i -th point p_i in P . For example, when $u = (0, 1, 1, 0, 1)^\top$ for $n = 5$ and $k = 3$, $S = \{p_2, p_3, p_5\}$.

The hypervolume subset selection problem (HSSP) [8], [16] is a special case of the ISSP using HV as \mathcal{I} . The HSSP has been well studied in the EMO community. In addition, the ISSPs using ϵ , R2, and IGD have also been addressed in [6], [17], and [18], respectively. The ISSP is an NP-hard combinatorial optimization problem as the same as a general subset selection problem [41]. Only for $d = 2$, the optimal

Algorithm 1: LS for the ISSP [3]

```

1 Initialize  $S \subset P$  randomly;
2 while There exists a pair of points that improves  $\mathcal{I}$  do
3   for  $s \in S$  do
4     for  $p \in P \setminus S$  do
5        $S' \leftarrow S \setminus \{s\} \cup \{p\}$ ;
6       if  $\mathcal{I}(S') > \mathcal{I}(S)$  then  $S \leftarrow S'$  ;

```

subset S^* of the ISSP using HV and ϵ can be found by dynamic programming [42].

The definition of the ISSP in equation (1) can be applied to any constrained multi-objective optimization [43], [44] when P consists of feasible non-dominated points. Therefore, conventional methods for the ISSP can perform subset selection of such P without any change. The existence of constraints could introduce a discontinuity in the PF, which is addressed in this paper.

D. Local Search for the ISSP

Algorithm 1 shows first-improvement LS for the ISSP presented in [3]. As mentioned in [3], a similar LS method was proposed in [20].

At the beginning of the search, a subset S is randomly initialized (line 1). Then, the following steps are repeatedly performed until there does not exist a pair of points that improves \mathcal{I} . For each iteration, a new subset is generated by swapping each point s in S and each unselected point p in $P \setminus S$ (line 5). Thus, LS in Algorithm 1 uses the 2-swap neighborhood. If the new subset S' is better than the current subset S in terms of \mathcal{I} , S is replaced with S' (line 6). Note that the final subset found by running LS is a local optimum for the 2-swap neighborhood.

Although the initial S is randomly generated (line 1), it can be set to a subset found by greedy search as demonstrated in [3]. In [14], the initial S is generated such that k points in S are uniformly distributed in the objective space V .

III. ANALYSIS OF THE DISTANCE BETWEEN TWO EXCHANGED POINTS

This section investigates the property of the ISSP to answer the following research question:

RQ1: How far apart are two points in a successful swap in LS?

Here, in lines 5–6 in Algorithm 1, we say that a swap of two points s and p in LS is successful when $\mathcal{I}(S') > \mathcal{I}(S)$. Recall that \mathcal{I} is to be maximized. Otherwise, we say that the swap is unsuccessful. Below, we focus on the distance between two points when their swap is successful or unsuccessful.

Let dist^s be the Euclidean distance between s and p for the successful swap. In addition, let dist^{us} be that for the unsuccessful swap. Note that both dist^s and dist^{us} are calculated in the objective space, not the solution space. Fig. 1 shows dist^s and dist^{us} for each evaluation of a subset by \mathcal{I} until the end of the search on the ISSP using the seven

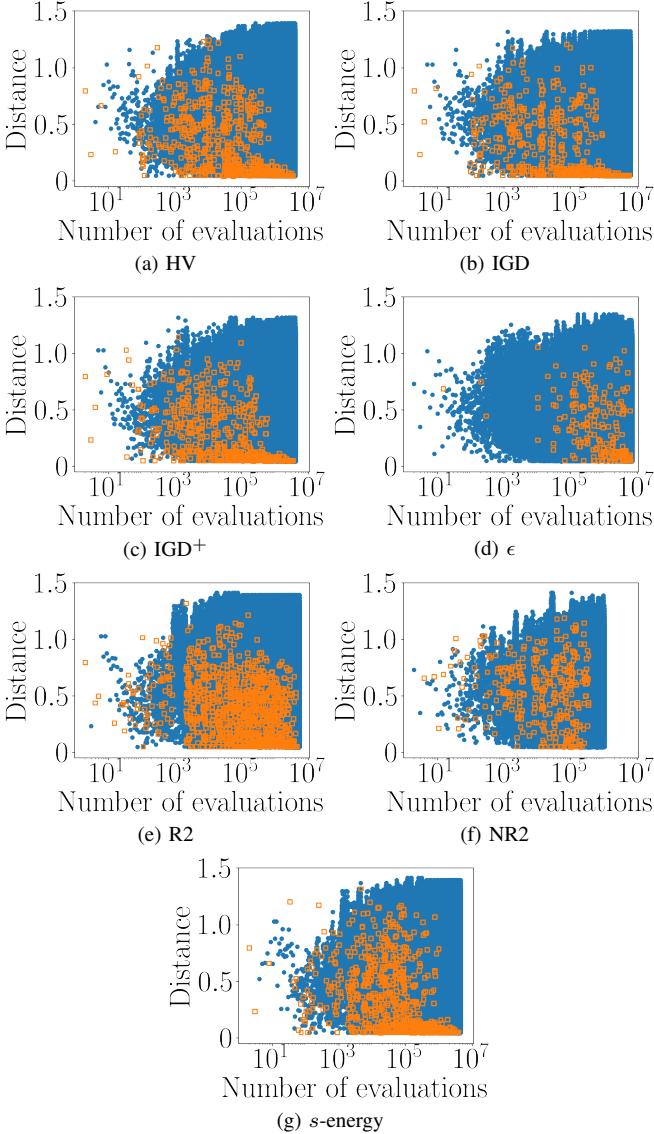


Fig. 1: Distribution of dist^s and dist^{us} values in a representative run of LS, where \square indicates a successful swap, and \bullet indicates an unsuccessful swap. The results are shown for the ISSPs using the seven quality indicators.

quality indicators (HV, IGD, IGD^+ , ϵ , R2, NR2, and s -energy) described in Section II-C. We set $n = 5000$ and $d = 4$. We describe the detail of the experimental setup in Section V later. Fig. 1 shows the results on the linear PF. In addition, Section S.1 investigates the relationship between the improvement of quality indicator values and dist^s . Section S.2 also investigates the relationship between the success rate of swapping two points and dist^s .

As shown in the results of HV in Fig. 1(a), dist^s values are relatively large within 10^6 evaluations. HV generally prefers a diverse distribution of points [32], [36], [37]. When a subset S is randomly initialized, the diversity of S is poor in most cases. For these reasons, swapping two points far away from each other is successful at an early stage. The previous study [14] proposed a variant of LS for the ISSP that swaps a point

in S and the farthest point from $P \setminus S$ for the first 50 iterations. Although the previous study [14] did not show the rationale of this strategy, it is consistent with our observation.

In contrast to the results at an early stage, as seen from Fig. 1(a), dist^s values are small for more than 10^6 evaluations, where $\text{dist}^s = 0.129$ even for the maximum case. This result indicates that the HV value of S generally cannot be improved by swapping two points far away from each other at a later stage of the search. This observation suggests that the number of evaluations of unpromising new subsets can be reduced by swapping a point in S only with its near points.

Although we discussed only the results for HV shown in Fig. 1(a), the results for IGD, IGD^+ , and s -energy are similar to the results for HV. In contrast, the results for R2, NR2, and ϵ are different from the results for HV. As described in Section II-B, the calculation of R2, NR2, and ϵ include the “min” and “max” operations. A contribution of each point in S can influence some quality indicators with the “sum” operation in a cumulative manner. In contrast, only d extreme points are likely to have an impact on some quality indicators with the “min” and “max” operations. In this case, other points do not contribute to the quality indicator value. Thus, the values of quality indicators that use the “min” and “max” operations may be improved or worsened only when changing points extremely. This is the reason why swapping two points far away from each other is effective on the ISSP using R2, NR2, and ϵ even at a later stage of the search, as shown in Figs. 1(d) and (e).

Figs. S.3 and S.4 in the supplementary file show the results for the nonconvex and convex PFs, respectively. Although we do not describe Figs. S.3 and S.4 in detail, these results are similar to those in Fig. 1, including the unexpected behavior of LS on the ISSP using R2, NR2, and ϵ (see the discussion above). Next, we fix the setting of the ISSP as follows: the PF = linear, $d = 4$, $n = 5000$, $k = 100$, and $\mathcal{I} = \text{HV}$. Then, we investigate the influence of PF, d , n , and k by varying each individually. Fig. S.5 shows the results for $\text{PF} \in \{\text{linear, convex, nonconvex, inverted-linear, inverted-convex, inverted-nonconvex}\}$. Fig. S.6 shows the results for $d \in \{2, 3, 4, 5, 6\}$. Fig. S.7 also shows the results for $n \in \{3000, 4000, 5000, 6000, 7000\}$. In addition, Fig. S.8 shows the results for $k \in \{50, 75, 100, 125, 150\}$. We do not describe these supplemental figures in detail here, but similar conclusions can be drawn from them. Finally, we performed five independent runs of LS, each with a different initial subset. The results in Fig. S.9 exhibit that all five runs show similar behavior. In summary, based on these results, we confirm that the conclusions in this section are universally applicable.

Answers to RQ1: Our results show that the distance between two points in a successful swap dist^s depends on the stage of the search and the type of quality indicator. On the ISSP using the HV, IGD, IGD⁺, and s -energy, while dist^s is large at an early stage, dist^s is small at a later stage. Thus, our results suggest that restricting the neighborhood of each point in S can possibly speed up LS on these ISSPs, at least at a later stage.

IV. PROPOSED METHOD

This section introduces the proposed candidate list strategy. First, Section IV-A describes LS with the candidate list strategy. Then, Sections IV-B and IV-C describe the nearest neighbor and random neighbor lists, respectively. Finally, Section IV-D proposes the two-phase candidate list strategy.

As noted in Section I, the candidate list strategy has not been previously studied in the context of the ISSP. Thus, the proposed candidate list strategy for the ISSP provides a new perspective. Note that the concept of the candidate list strategy itself is general and not novel. No work can claim that using the candidate list strategy itself is the original contribution. Instead, the main point is how to make use of such a general idea on a particular problem domain. In fact, how to restrict the neighborhood is not obvious on an unseen problem, including the ISSP. We designed the proposed candidate list strategy based on the analysis in Section III, where the analysis is another key contribution of this paper.

A. Local search with the candidate list strategy

Algorithm 2 shows LS with a candidate list. The candidate list L_p is a set of l possible points that are swapped with $p \in P$. Here, l is the size of the candidate list. Thus, there are n candidate lists L_{p_1}, \dots, L_{p_n} for n points p_1, \dots, p_n , respectively. The only difference between Algorithms 1 and 2 lies in line 4. For each selected point s in the current subset S , Algorithm 1 swaps s and an unselected point p in P . In contrast, Algorithm 2 swaps s and p only in the corresponding candidate list L_s .

Recall that n is the size of P , and k is the size of S . For each iteration, LS in Algorithm 1 swaps selected k points in S for unselected $n-k$ points in $P \setminus S$. Thus, each LS iteration requires $k(n-k)$ evaluations of a quality indicator \mathcal{I} [3]. In contrast, for each iteration, LS with the candidate list strategy in Algorithm 2 swaps k points in S only for l points from the corresponding candidate lists in the maximum case. This means that LS with the candidate list strategy scans only the neighborhood of size kl even in the worst case. In summary, the candidate list strategy can significantly reduce the number of evaluations of \mathcal{I} at each iteration from $k(n-k)$ to kl . However, unlike Algorithm 1, Algorithm 2 needs to construct n candidate lists for n points in P at the beginning of the search. Thus, Algorithm 2 requires additional time and space costs.

The remaining question is how to generate the candidate list. This paper proposes two types of candidate lists: the nearest neighbor and random neighbor lists. First, Section IV-B introduces the nearest neighbor list that includes only

Algorithm 2: LS with the proposed candidate list strategy

```

1 Initialize  $S \subset P$  randomly;
2 while There exists a pair of points that improves  $\mathcal{I}$  do
3   for  $s \in S$  do
4     for  $p \in L_s \setminus S$  do
5        $S' \leftarrow S \setminus \{s\} \cup \{p\}$ ;
6       if  $\mathcal{I}(S') > \mathcal{I}(S)$  then  $S \leftarrow S'$  ;

```

Algorithm 3: A method for generating L^N

```

1 for  $p \in P$  do
2    $L_p^N \leftarrow \emptyset$ ;
3   while  $|L_p^N| < l$  do
4      $L_p^N \leftarrow L_p^N \cup \left\{ \underset{q \in P \setminus \{p\} \setminus L_p^N}{\operatorname{argmin}} \text{dist}(p, q) \right\}$ ;

```

the l nearest points in the objective space. We emphasize that the term “near” is in the objective space, not in the search space of the ISSP.³ Then, Section IV-C introduces the random neighbor list that includes only l randomly selected points at the beginning of the search. If the aim is only to reduce the computational cost of LS, this can be achieved by using any neighbor list. However, this paper aims to reduce the computational cost of LS without significantly compromising the quality of subsets on various ISSPs. For this purpose, the two neighbor lists are necessary.

B. Nearest neighbor list

For each $p \in P$, the nearest neighbor list L_p^N is a set of the l nearest points to p in the objective space. Algorithm 3 shows a method for generating the nearest neighbor list. For each $p \in P$, L_p^N is initialized to an empty set (line 2). Then, the nearest point to p is repeatedly added to L_p^N until $|L_p^N| = l$ (line 4). Here, $\text{dist}(p, q)$ represents the Euclidean distance between p and q in the objective space.

Fig. 2 shows examples of distributions of points on continuous and discontinuous linear PFs. In the example of Fig. 2(a), the current subset S is $\{p_1, p_4, p_6\}$. When $l = 2$, the nearest neighbor lists of p_1 , p_4 , and p_6 are $L_{p_1}^N = \{p_2, p_3\}$, $L_{p_4}^N = \{p_3, p_5\}$, and $L_{p_6}^N = \{p_5, p_7\}$, respectively. Let us now consider the swap operation for p_1 . In LS in Algorithm 1, p_1 can be swapped with the four points (p_2 , p_3 , p_5 , and p_7). In contrast, in LS with the nearest neighbor list in Algorithm 2, p_1 can be swapped only with the two nearest points (p_2 , p_3).

As observed in Section III, swapping two points far away from each other in the objective space is unlikely to improve the quality indicator value of S at a later stage of the search. In contrast, Algorithm 2 using the nearest neighbor list L^N swaps only two points close to each other in the objective space. Thus, we believe that Algorithm 2 using L^N can

³For example, as mentioned in Section II-C, a solution of the ISSP can be represented by an n -dimensional 0-1 vector. However, we are not interested in how near two 0-1 vectors are in terms of the Hamming distance.

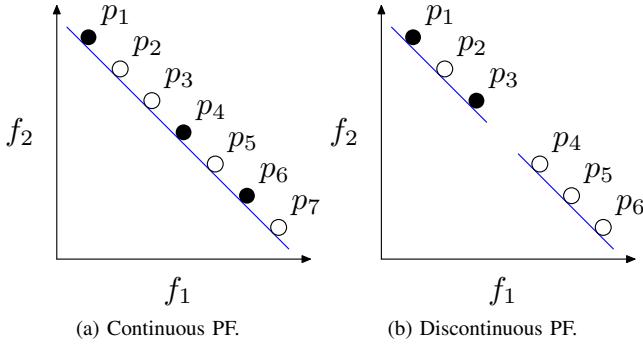


Fig. 2: Examples of distributions of points on continuous and discontinuous PFs.

save unpromising swaps. Note that Algorithm 2 using L^N can indirectly swap two points far away from each other by repeatedly swapping two points close to each other on a continuous PF. For example, when $l = 2$ in Fig. 2(a), a swap of p_4 and $p_7 \notin L_{p_4}^N$ can be performed by swapping 1) p_4 and $p_5 \in L_{p_4}^N$, 2) p_6 and $p_7 \in L_{p_6}^N$, and 3) p_5 and $p_6 \in L_{p_5}^N$.

The time and space complexities of Algorithm 3 are $O((d + \log n)n^2)$ and $O((l + d)n)$, respectively. First, the distance calculation $\text{dist}(p, q)$ is needed for each $p \in P$ and $q \in P \setminus \{p\}$. Next, for each $p \in P$, these values are sorted to find l nearest points from p . Note that the time complexity can be reduced to $O(dn^2)$ by simply selecting l smallest values.

In our experiments, constructing the nearest neighbor list requires only a few seconds for $n \leq 10^4$ even in the worst case. We also observed that the computation time for constructing the nearest neighbor list is much smaller than that for the search by LS. Thus, the construction of the nearest neighbor list is computationally cheap in practice, except for a special case (e.g., stopping LS immediately after list construction).

C. Random neighbor list

For each $p \in P$, the random neighbor list L_p^R consists a set of l randomly selected points from $P \setminus \{p\}$. The random neighbor list is generated only once at the beginning of the search. As noted in Section IV-B, using the nearest neighbor list L^N allows LS to perform an indirect swap of two points far away from each other on a *continuous* PF. However, this is not always true for a *discontinuous* PF.

The PF in Fig. 2(b) consists of two subsets. When $S = \{p_1, p_3\}$ and $l = 2$, $L_{p_1}^N = \{p_2, p_3\}$ and $L_{p_3}^N = \{p_1, p_2\}$. In this case, LS using the nearest neighbor list cannot swap $p \in \{p_1, p_3\}$ for $q \in \{p_4, p_5, p_6\}$. In other words, LS using the nearest neighbor list can swap points only on the same subset of the PF. For this reason, Algorithm 2 using the nearest neighbor list is likely to perform worse than Algorithm 1 on ISSP instances with discontinuous PFs in terms of the quality of subsets. In contrast, Algorithm 2 using the random neighbor list can swap points on different subsets of the PF. Therefore, we believe that the random neighbor list is effective on an ISSP instance with a discontinuous PF.

The expected time complexity of randomly selecting l unique points from P is only $O(ln)$. Thus, the time complexity

Algorithm 4: LS using the two neighbor lists

- 1 Initialize S randomly;
 - 2 $L^R \leftarrow$ Generate the random neighbor list of P ;
 - 3 $S \leftarrow$ Apply Algorithm 2 using L^R to S ;
 - 4 $L^N \leftarrow$ Generate the nearest neighbor list of P ;
 - 5 $S \leftarrow$ Apply Algorithm 2 using L^N to S ;

of constructing the random neighbor list is significantly lower than that of constructing the nearest neighbor list described in Section IV-B.

D. Sequential use of the two neighbor lists

As discussed in Section IV-C, LS using the nearest neighbor list is likely to perform poorly on an ISSP instance with a discontinuous PF. However, we believe that LS using the random neighbor list can achieve only a poor-quality subset. This is simply because this version of LS can swap only randomly selected points at the beginning of the search. To address these issues, this section considers the sequential use of the random neighbor and nearest neighbor lists. It is expected that the drawbacks of the two neighbor lists complement each other by using them sequentially. Note that the use of multiple neighbor lists has been well studied in the context of variable neighborhood search [45]. Our contribution here is to propose an efficient way to combine two specific lists for the ISSP.

Algorithm 4 shows LS using the two neighbor lists. First, the subset S is randomly initialized (line 1). Then, Algorithm 2 using the random neighbor list L^R is applied to S (lines 2–3). Finally, S is further improved by Algorithm 2 using the nearest neighbor list L^N (lines 4–5). Here, in Algorithm 4, S is not re-initialized at line 1 in Algorithm 2. Since Algorithm 4 uses the two lists, it requires the total time complexity of constructing them described in Sections IV-B and IV-C, respectively.

Our results in Section III showed that swapping two points far away from each other can improve the quality indicator value of a subset at an early stage in LS (Algorithm 1). Our results also showed that swapping two points close to each other is effective at a later stage. We believe that Algorithm 4 can behave similarly to these successful swapping operations in Algorithm 1. The random neighbor list allows LS to swap two points far away from each other. In contrast, the nearest neighbor list restricts LS to swap only two points close to each other.

We restrict the neighborhood at all stages by using the two neighbor lists. This is to reduce the number of examined solutions that do not lead to improvements. It may be possible not to use the nearest neighbor list at “the early stage” and to use it at “the later stage”. However, this raises the question of how to define “the early stage”. For example, it is possible to switch the search strategy after u iterations. However, the best u value clearly depends on a problem, and finding it on a real-world problem is difficult. Designing an adaptive switching strategy can be an avenue for future research.

V. EXPERIMENTAL SETUP

This section describes the experimental setup. We conducted all experiments on a workstation with an AMD Ryzen Threadripper PRO 3975WX (32-core, 3.5GHz) processor and 512GB of RAM using Ubuntu 22.04. We implemented all LS methods in C++.⁴ The programs were compiled using GNU C++ compiler 11.1.0 with optimization level O2.

This paper considers the ISSPs of the following seven quality indicators: HV, IGD, IGD⁺, R2, NR2, ϵ , and s -energy. The pagmo [46] implementation of the WFG algorithm [47] was used to calculate HV. For HV and NR2, the reference point $r \in \mathbb{R}^d$ was set to $r = (1.1, \dots, 1.1)^\top$. We set the size of the reference point set in IGD and IGD⁺ to 1 000. We also used a weight vector set of size 1 000 for R2 and NR2.

As in [15], k was set to 100 unless otherwise noted. Section VI shows the results on the linear continuous PF in DTLZ1 and discontinuous PF in DTLZ7. In our preliminary experiment, as in [15], we used the following six continuous PFs: a linear PF (DTLZ1), concave PF (DTLZ2), convex PF (convDTLZ2), and their inverted versions. However, our preliminary results showed that the results for the six PFs are similar. For this reason, this paper shows the results only for the linear PF.

We generated a non-dominated point set P of size n for each PF. First, P was uniformly generated on a linear PF using the method proposed in [48], where $\sum_{i=1}^d p_i = 1$ for each p in P . Then, P was translated for each PF using the method presented in [49]. We set the point set size n to 1 000, 2 000, ..., 10 000. We also set the number of objectives d to 2, 4, and 6. This work used neither the maximum number of subset evaluations by a quality indicator nor the cut-off time as a termination condition. Instead, all LS methods were terminated when there does not exist a pair of points that improves a given quality indicator \mathcal{I} . We performed 31 runs of each LS method.

Section VI investigates the performance of the following four LS methods:

LS Conventional LS (Section II),

LS-N LS using the nearest neighbor list (Section IV-B),

LS-R LS using the random neighbor list (Section IV-C),

LS-RN LS using the two neighbor lists (Section IV-D).

Let l^N be the size of the nearest neighbor list L^N . Let also l^R be the size of the random neighbor list L^R . This work used $l^N = 40$ for LS-N, $l^R = 40$ for LS-R, and $l^N = l^R = 20$ for LS-RN ($l^N + l^R = 40$) unless otherwise noted.

VI. RESULTS

This section describes our analysis results. Through experiments, we address the following four research questions (RQ2–RQ5) in Sections VI-A–VI-D, respectively. Section VI-E shows further investigations.

RQ2: Can the proposed candidate list strategy speed up LS on the ISSP with a continuous PF?

RQ3: How does the performance of the candidate list strategy scale with respect to the number of objectives d and the subset size k ?

RQ4: Can the proposed candidate list strategy handle a discontinuous PF?

RQ5: How does the list size l influence the effectiveness of the proposed candidate list strategy?

A. Effectiveness of the candidate list strategy

First, we investigate how the performance of LS using the proposed candidate list strategy scales with respect to n . For each run of an LS method, we measure *i*) a quality indicator value $\mathcal{I}(S^{\text{bsf}})$ of the best subset found so far S^{bsf} , *ii*) the number of subset evaluations by \mathcal{I} , and *iii*) the wall-clock time of the run.

1) Quality of subsets found by LS: Table I shows the results of the four LS methods (LS, LS-N, LS-R, and LS-RN) on the linear PF in terms of the quality indicator value $\mathcal{I}(S)$ on the ISSP using HV, IGD, R2, and ϵ . Table S.1 shows the results for all seven quality indicators, including IGD⁺, NR2, and s -energy. Table S.1 also shows the standard deviation of $\mathcal{I}(S^{\text{bsf}})$. Since Table S.1 is similar to Table I, we do not describe Table S.1. As seen from Table I, the four LS methods find subsets with very similar quality.

Below, we discuss the relative performance of LS-N, LS-R, and LS-RN compared to conventional LS in terms of the quality of the subsets. Let $I_{\text{LS}}^{\text{mean}}$ be the mean of the quality indicator values of the best subsets found so far by an LS method over 31 runs. We calculated the relative error for conventional LS and the three proposed methods (LS-N, LS-R, and LS-RN) as follows: $(I_{\text{LS}}^{\text{mean}} - I_{\text{LS}'}^{\text{mean}}) / |I_{\text{LS}}^{\text{mean}}|$, where LS' is either LS-N, LS-R, or LS-RN. A large relative error suggests that the corresponding LS method finds worse subsets than the conventional LS method. Note that a negative relative error value can be observed if the corresponding LS method finds better subsets than the conventional LS method. We conducted the Wilcoxon rank-sum test with $\alpha < 0.05$. Table S.2 shows the results of the statistical tests for LS-RN with respect to LS-N and LS-R. We do not explain Table S.2 in detail, but Table S.2 shows that LS-RN performs better than or is comparable to LS-N and LS-R in most cases.

As shown in Tables I (a), (b), and (c), LS-N and LS-RN perform significantly worse than conventional LS on the ISSP with HV, IGD, and R2. However, the relative error is 0.897% only in the maximum case. Here, the previous study [17] concluded that the relative error in the range 0.0677% – 3.5109% is considered “very small”. Based on this, our results indicate that LS-N and LS-RN can find well-approximated subsets. As expected, LS-R is outperformed by LS, LS-N, and LS-RN. This outcome is natural because swapping only randomly selected points is not effective. As discussed in Section IV-D, swapping two points close to each other is more effective than swapping randomly selected points on a continuous PF in terms of the quality of the subsets. However, the relative error for LS-R is 4.87% even in the maximum case.

As seen from Table I(d), the results for ϵ are different from those for other quality indicators. LS-N, LS-R, and LS-RN achieve significantly worse subsets than LS in terms of ϵ . For example, the relative error for LS-N is 27% in the maximum case. LS-R obtains a better quality indicator value than LS-N,

⁴The implementation is available at https://github.com/rogj52/issp_ls_clist

TABLE I: Comparison of the four LS methods on the ISSP instances with the linear PF and HV, IGD, R2, and ϵ . The average quality indicator values of the subsets found by each LS method are shown. The symbols + and - indicate that the corresponding LS with the candidate list strategy (LS-N, LS-R, or LS-RN) performs significantly better and significantly worse than conventional LS according to the Wilcoxon rank-sum test with $\alpha < 0.05$, respectively. The symbol \approx indicates no significant difference between the results of the corresponding LS with the candidate list strategy and conventional LS.

(a) HV					
n	LS	LS-N	LS-R	LS-RN	
1K	1.37e+00	1.37e+00 - (6.42e-03%)	1.37e+00 - (7.56e-02%)	1.37e+00 - (8.36e-03%)	
2K	1.37e+00	1.37e+00 - (3.57e-03%)	1.37e+00 - (9.92e-02%)	1.37e+00 - (5.14e-03%)	
3K	1.37e+00	1.37e+00 - (6.44e-03%)	1.37e+00 - (1.08e-01%)	1.37e+00 - (6.96e-03%)	
4K	1.37e+00	1.37e+00 - (6.10e-03%)	1.37e+00 - (1.20e-01%)	1.37e+00 - (6.38e-03%)	
5K	1.37e+00	1.37e+00 - (7.72e-03%)	1.37e+00 - (1.25e-01%)	1.37e+00 - (6.19e-03%)	
6K	1.37e+00	1.37e+00 - (7.19e-03%)	1.37e+00 - (1.36e-01%)	1.37e+00 - (7.54e-03%)	
7K	1.37e+00	1.37e+00 - (5.78e-03%)	1.37e+00 - (1.38e-01%)	1.37e+00 - (7.17e-03%)	
8K	1.37e+00	1.37e+00 - (6.00e-03%)	1.37e+00 - (1.41e-01%)	1.37e+00 - (8.97e-03%)	
9K	1.37e+00	1.37e+00 - (6.95e-03%)	1.37e+00 - (1.40e-01%)	1.37e+00 - (9.03e-03%)	
10K	1.37e+00	1.37e+00 - (8.56e-03%)	1.37e+00 - (1.40e-01%)	1.37e+00 - (8.53e-03%)	
(b) IGD					
n	LS	LS-N	LS-R	LS-RN	
1K	8.07e-02	8.10e-02 - (4.25e-01%)	8.24e-02 - (2.14e+00%)	8.08e-02 - (1.52e-01%)	
2K	8.06e-02	8.09e-02 - (3.24e-01%)	8.31e-02 - (3.06e+00%)	8.08e-02 - (2.02e-01%)	
3K	7.98e-02	8.00e-02 - (2.33e-01%)	8.31e-02 - (4.09e+00%)	8.00e-02 - (2.91e-01%)	
4K	7.96e-02	7.98e-02 - (2.99e-01%)	8.30e-02 - (4.31e+00%)	7.97e-02 - (1.99e-01%)	
5K	7.94e-02	7.96e-02 - (3.66e-01%)	8.28e-02 - (4.34e+00%)	7.96e-02 - (2.53e-01%)	
6K	7.92e-02	7.95e-02 - (3.74e-01%)	8.28e-02 - (4.51e+00%)	7.94e-02 - (2.04e-01%)	
7K	7.92e-02	7.94e-02 - (2.84e-01%)	8.27e-02 - (4.47e+00%)	7.93e-02 - (1.83e-01%)	
8K	7.91e-02	7.94e-02 - (3.30e-01%)	8.28e-02 - (4.59e+00%)	7.93e-02 - (1.91e-01%)	
9K	7.90e-02	7.93e-02 - (3.67e-01%)	8.27e-02 - (4.72e+00%)	7.92e-02 - (2.89e-01%)	
10K	7.90e-02	7.93e-02 - (3.70e-01%)	8.28e-02 - (4.87e+00%)	7.92e-02 - (2.52e-01%)	
(c) R2					
n	LS	LS-N	LS-R	LS-RN	
1K	2.17e-02	2.17e-02 - (1.19e-01%)	2.18e-02 - (4.96e-01%)	2.17e-02 - (1.08e-01%)	
2K	2.15e-02	2.16e-02 - (3.12e-01%)	2.17e-02 - (1.00e+00%)	2.16e-02 - (2.95e-01%)	
3K	2.13e-02	2.14e-02 - (3.52e-01%)	2.18e-02 - (1.98e+00%)	2.14e-02 - (3.55e-01%)	
4K	2.13e-02	2.14e-02 - (4.63e-01%)	2.19e-02 - (2.78e+00%)	2.14e-02 - (4.31e-01%)	
5K	2.13e-02	2.14e-02 - (5.38e-01%)	2.19e-02 - (2.76e+00%)	2.14e-02 - (4.79e-01%)	
6K	2.12e-02	2.14e-02 - (5.99e-01%)	2.19e-02 - (3.32e+00%)	2.13e-02 - (5.62e-01%)	
7K	2.12e-02	2.13e-02 - (6.77e-01%)	2.22e-02 - (4.47e+00%)	2.13e-02 - (6.22e-01%)	
8K	2.12e-02	2.13e-02 - (6.88e-01%)	2.20e-02 - (3.85e+00%)	2.14e-02 - (7.65e-01%)	
9K	2.12e-02	2.13e-02 - (7.43e-01%)	2.21e-02 - (4.46e+00%)	2.13e-02 - (6.23e-01%)	
10K	2.12e-02	2.13e-02 - (8.97e-01%)	2.21e-02 - (4.57e+00%)	2.13e-02 - (8.18e-01%)	
(d) ϵ					
n	LS	LS-N	LS-R	LS-RN	
1K	8.22e-02	1.04e-01 - (2.71e+01%)	8.11e-02 + (-1.31e+00%)	8.07e-02 + (-1.90e+00%)	
2K	9.30e-02	9.78e-02 - (5.24e+00%)	9.48e-02 \approx (1.94e+00%)	9.58e-02 \approx (3.05e+00%)	
3K	8.67e-02	9.30e-02 - (7.34e+00%)	8.57e-02 \approx (-1.09e+00%)	8.43e-02 \approx (-2.71e+00%)	
4K	7.87e-02	8.74e-02 - (1.11e+01%)	8.50e-02 - (7.97e+00%)	7.88e-02 \approx (1.94e-01%)	
5K	7.28e-02	8.50e-02 - (1.68e+01%)	7.97e-02 - (9.57e+00%)	7.72e-02 - (6.14e+00%)	
6K	7.36e-02	9.81e-02 - (3.33e+01%)	8.55e-02 - (1.61e+01%)	8.48e-02 - (1.52e+01%)	
7K	7.74e-02	9.29e-02 - (1.99e+01%)	9.01e-02 - (1.63e+01%)	8.39e-02 \approx (8.36e+00%)	
8K	7.75e-02	8.84e-02 - (1.41e+01%)	8.77e-02 - (1.31e+01%)	8.46e-02 \approx (9.14e+00%)	
9K	7.25e-02	8.78e-02 - (2.12e+01%)	8.79e-02 - (2.13e+01%)	8.23e-02 - (1.36e+01%)	
10K	7.44e-02	9.13e-02 - (2.27e+01%)	8.60e-02 - (1.56e+01%)	8.27e-02 - (1.11e+01%)	

except for the results for $n = 5\,000$. In contrast, LS-RN finds better subsets than both LS-N and LS-R in most cases. This result indicates that the performance of LS-RN is less sensitive to the type of quality indicator compared to LS-N and LS-R.

2) *Speed improvement by the candidate list:* Fig. 3 shows the results of each LS method on the ISSP with the linear PF using HV, IGD, R2, and ϵ . Fig. 3 shows the average of the number of subset evaluations by \mathcal{I} and the wall-clock time of the run. Fig. S.10 shows the results for IGD⁺, NR2, and s -energy. Since the results for IGD⁺, NR2, and s -energy are similar to those for IGD, they are not shown in Fig. 3.

As seen from the results of LS-N, LS-R, and LS-RN in Figs. 3(a)–(d), the number of subset evaluations is significantly

reduced by using the candidate list strategy for all quality indicators. For example, on the ISSP using HV, the reduction rate of the number of subset evaluations by LS-RN is 96.5% for $n = 1\,000$ and 99.6% for $n = 10\,000$. Thus, the results show that the candidate list strategy can speed up LS in terms of the number of subset evaluations, especially for the ISSP with a large n . Except for ϵ , the results of LS-N, LS-R, and LS-RN in Fig. 3 are similar. This is because the number of examined subsets at each iteration of LS-N, LS-R, and LS-RN is almost the same due to the property of the candidate list strategy. While LS scans the neighborhood of size $k(n - k)$ at each iteration, LS-N, LS-R, and LS-RN scan only the neighborhood of size kl , even in the worst case. Recall that

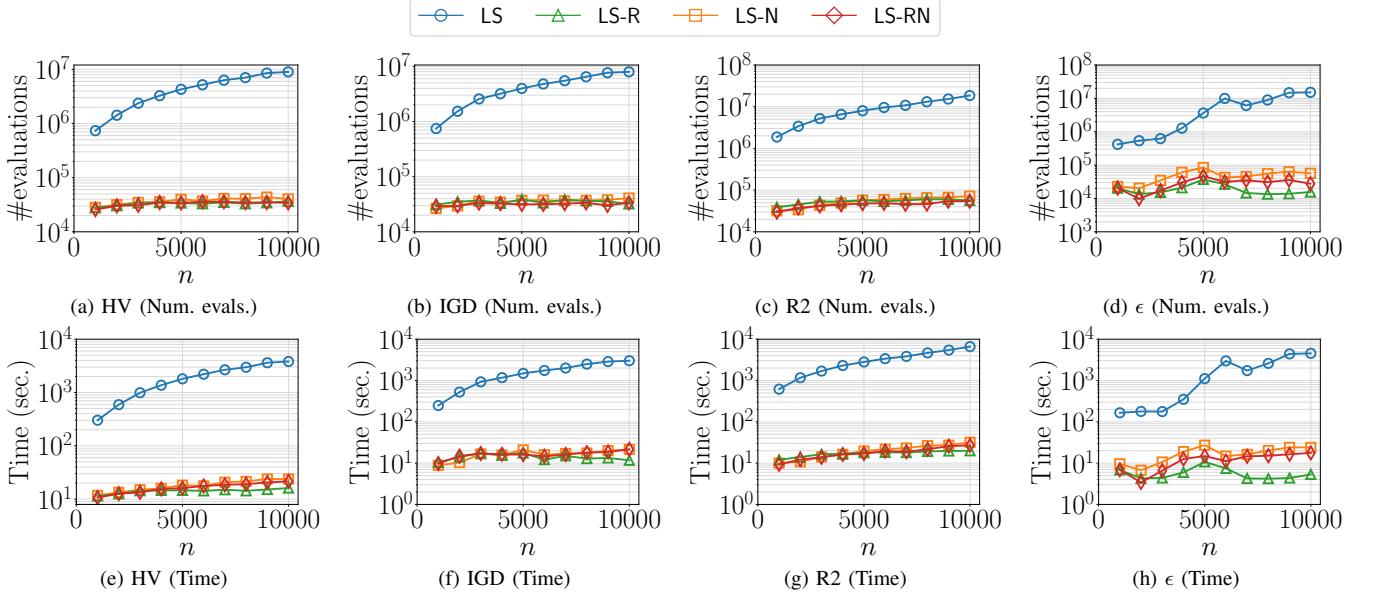


Fig. 3: Results of the four LS methods on the ISSP with the linear PF using HV, IGD, R2, and ϵ . The average number of subset evaluations by \mathcal{I} and the wall-clock time of the run are shown in figures (a)–(d) and (e)–(h), respectively.

LS-N, LS-R, and LS-RN use the same l . As discussed in Section III, ϵ has a different property compared to other quality indicators. This makes the difference between the results of the three LS methods for ϵ .

As shown in Figs. 3(e)–(h), the candidate list strategy can also speed up LS in terms of the computation time. For example, LS-RN is approximately 29–175 times faster than LS on the ISSP using HV in terms of the computation time. As seen from Fig. 3, LS-RN is slightly faster than LS-N in terms of both number of subset evaluations and computation time. This is due to the effect of the first LS-R phase in LS-RN.

In summary, our results show that using the candidate list strategy slightly degrades the quality of subsets found by LS but drastically reduces the computational cost. It should be noticed that LS with the candidate list strategy can find a better subset through repeatedly performing restarts.

Answers to RQ2: Our results of LS-RN and LS-N demonstrate that the candidate list strategy can drastically speed up LS in terms of both number of subset evaluations and computation time without significantly compromising the quality of subsets on the ISSP using HV, IGD, IGD⁺, R2, NR2, and s -energy. The candidate list strategy is effective especially when n is large. LS-RN performs better than LS-N in terms of both quality of subsets and computational cost. However, both LS-RN and LS-N perform poorly in terms of the quality of subsets on the ISSP using ϵ due to its property observed in Section III. Unlike LS-N and LS-RN, LS-R achieves only poor subsets in most cases. This observation suggests the importance of the nearest neighbor list.

B. Scalability of the candidate list strategy to d and k

1) *Scalability to the number of objectives d :* Fig. 4 shows the average results of the four LS methods on the ISSP with $n = 5000$, $k = 100$, and $d \in \{2, 3, 4, 5, 6\}$ using HV and ϵ . Since the results are consistent across values of n , we show only those for $n = 5000$. For all seven quality indicators, Table S.3 shows the results in terms of the quality indicator value. In addition, Fig. S.11 shows the results in terms of the number of subset evaluations and the computation time. We do not describe Table S.3 in detail, but Table S.3 is similar to Table I. Thus, the results show that LS-RN and LS-N find slightly worse subsets than the conventional LS in most cases, except on the ISSP using ϵ . We also do not describe Fig. S.11 in detail, but Fig. S.11 is similar to Fig. 4.

As seen from Fig. 4, LS-R terminates the search earlier than LS-N and LS-RN for smaller values of d . This suggests that LS-R achieves fewer successful point swaps compared to LS-N and LS-RN. In Fig. 4, LS-R, LS-N, and LS-RN perform similarly in terms of the number of subset evaluations as d increases. Contrary to intuition, the number of subset evaluations in all four LS methods decreases as d increases. As shown in Figs. S.11(a)–(g), similar results can be observed on the ISSPs using other quality indicators. Similar to Figs. 3(e)–(h), Figs. S.11(h)–(n) show the results regarding the computation time. The results for HV in Fig. S.11(h) show that all four LS methods require higher computation time as d increases. However, this observation is not inconsistent with Fig. 4(a). This is because the computation time of HV grows exponentially with d .

2) *Scalability to the subset size k :* Fig. 5 shows the average results of the four LS methods on the ISSP with $d = 4$, $n = 5000$, and $k \in \{50, 60, \dots, 150\}$ using HV and ϵ in terms of the number of subset evaluations. Fig. S.12 shows the results for other quality indicators. The results for IGD, IGD⁺, R2,

TABLE II: Comparison of the four LS methods on the ISSP instances with the discontinuous PF using HV. The average quality indicator values of subsets found by the four methods are shown.

n	LS	LS-N	LS-R	LS-RN	
1K	5.90e-01	5.83e-01	- (1.09e+00%)	5.88e-01	- (3.55e-01%)
2K	5.94e-01	5.89e-01	- (8.84e-01%)	5.91e-01	- (5.25e-01%)
3K	5.94e-01	5.89e-01	- (8.53e-01%)	5.91e-01	- (5.61e-01%)
4K	5.94e-01	5.86e-01	- (1.29e+00%)	5.90e-01	- (6.72e-01%)
4K	5.95e-01	5.89e-01	- (9.63e-01%)	5.91e-01	- (6.36e-01%)
5K	5.94e-01	5.87e-01	- (1.20e+00%)	5.90e-01	- (7.16e-01%)
6K	5.95e-01	5.89e-01	- (1.05e+00%)	5.91e-01	- (6.77e-01%)
8K	5.94e-01	5.87e-01	- (1.27e+00%)	5.90e-01	- (7.46e-01%)
9K	5.95e-01	5.89e-01	- (1.02e+00%)	5.91e-01	- (7.23e-01%)
10K	5.95e-01	5.90e-01	- (7.39e-01%)	5.91e-01	- (7.13e-01%)

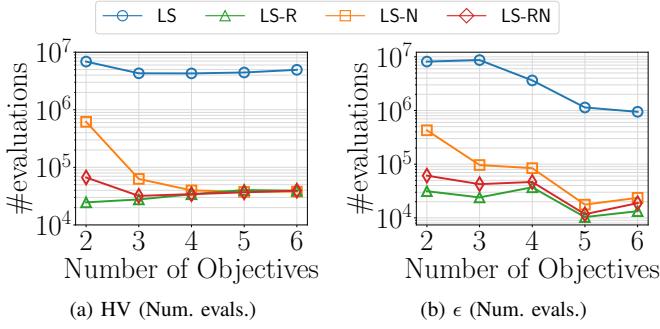


Fig. 4: Scalability of the four LS methods to the number of objectives d on the ISSP with the linear PF using HV and ϵ .

Answers to RQ3: Our results show that LS-RN is well-scalable to the number of objectives d and the subset size k . Our results suggest that LS-RN is effective especially for a large d . LS-RN can significantly reduce the computation time especially on the ISSP using HV. We observed that LS-N performs worse than LS-R in terms of the number of subset evaluations and computation time when d is small. We also found that these differences between LS-N, LS-R, and LS-RN become small as d increases. Overall, our results show that LS-RN generally obtains a good subset for any \mathcal{I} , d , and k compared to LS-R and LS-N. The results indicate the importance of using the two neighbor lists sequentially.

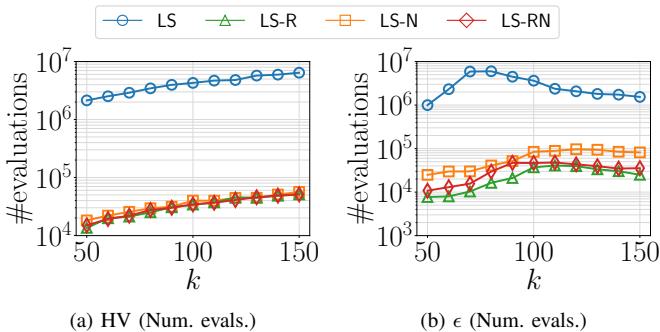


Fig. 5: Scalability of the four LS methods to the subset size k on the ISSP with the linear PF using HV and ϵ .

NR2, and s -energy are similar to those for HV in Fig. 5. Table S.4 shows the quality of subsets for $k \in \{50, 60, \dots, 150\}$. Similar to Table I, Table S.4 shows that LS-N and LS-RN find subsets with acceptable quality.

As seen from Fig. 5(a), the number of subset evaluations in all four LS methods increases as k increases. However, Fig. 5(a) demonstrates that the candidate list strategy significantly reduces the number of subset evaluations in LS for any k . As shown in Fig. 5(b), for ϵ , the number of subset evaluations in LS decreases as k increases from 70 to 150. LS with the candidate list strategy also shows a similar trend. However, for any k , the number of subset evaluations in LS with the candidate list strategy is significantly smaller than that in conventional LS.

C. Effectiveness of the candidate list strategy on the discontinuous PF

Unlike Section VI-A, we here investigate the effectiveness of the candidate list strategy on an ISSP with a discontinuous PF. We used the discontinuous PF of the DTLZ7 problem [24]. We normalized the PF of the DTLZ problem into $[0, 1]^d$ so that the scale of objectives is the same as in other PFs. We uniformly generated Pareto optimal points on the PF by using the method proposed in [49]. Due to the combinatorial property of this generation method, we set the point set size n to 1 000, 2 197, 3 375, 4 096, 4 913, 5 832, 6 859, 8 000, 9 261, and 10 648. Here, we set d to 4. Table S.6 shows the results of the statistical tests for LS-RN with respect to LS-N and LS-R. Although we do not explain Table S.6 in detail, it shows that LS-RN performs significantly better than LS-N and LS-R, except for the results for ϵ .

Fig. 6 shows the average results of the four LS methods on the ISSP with the discontinuous PF using HV, IGD, R2, and ϵ in terms of the number of subset evaluations and computation time. We show the results for IGD^+ , NR2, and s -energy in Fig. S.13 for the same reason as in Fig. 3. Fig. 6 is similar to Fig. 3. Thus, as shown in Fig. 6, the candidate list strategy can speed up LS even on the discontinuous PF.

Table II shows the comparison of the four LS methods in terms of the quality of the subsets on the ISSP using HV. Table S.5 shows the results for other quality indicators, but they are similar to Table II. As shown in Table II, LS-N performs significantly worse than conventional LS in terms of the quality of subsets. For example, as shown in Table I(a), the relative error of LS-N compared to LS is only $6.42 \times 10^{-3}\%$ on the ISSP with the linear PF and $n = 1 000$ using HV. In

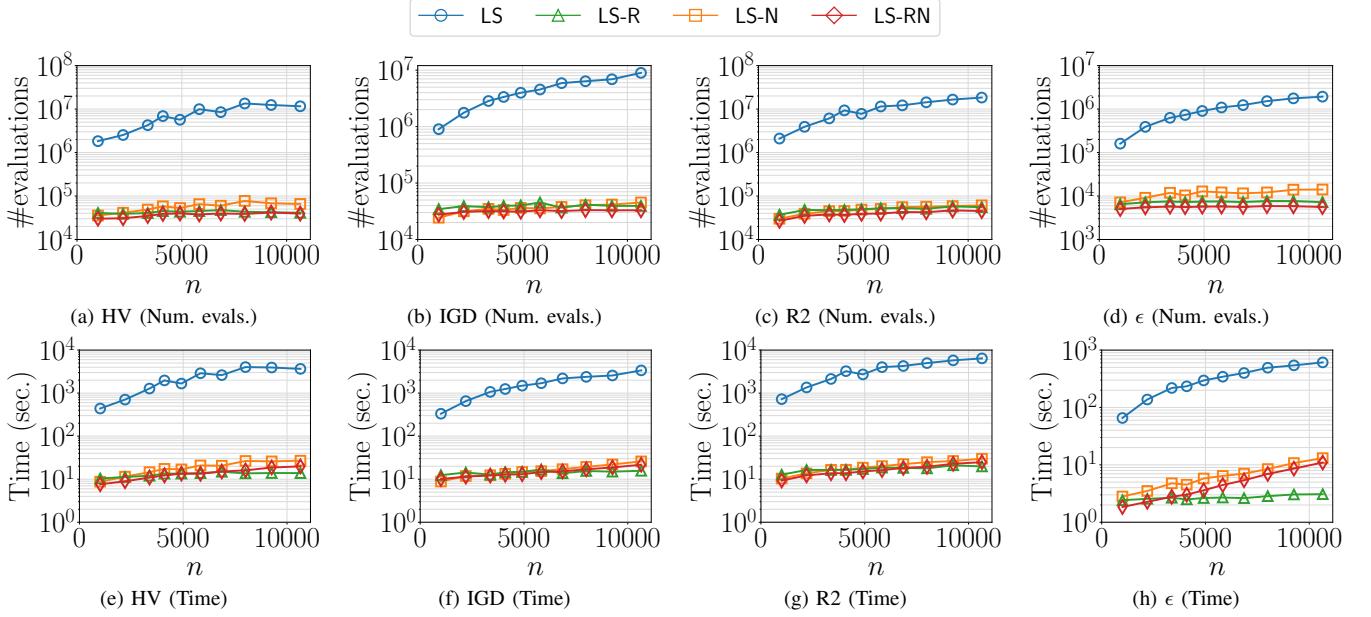


Fig. 6: Results of the four LS methods on the ISSP with the discontinuous PF using HV, IGD, R2, and ϵ .

contrast, that is 1.09% on the ISSP with the discontinuous PF. Interestingly, LS-N obtains worse subsets than LS-R on the ISSP using HV, where similar results are found on the ISSP using R2 and ϵ .

As discussed in Section IV-C, the poor performance of LS-N for the discontinuous PF comes from the difficulty in swapping points on different subsets of the PF. LS-RN addresses this issue of LS-N by using the random neighbor and the nearest neighbor lists sequentially. As seen from Table II and Fig. 6, LS-RN finds better subsets than LS-N while reducing the computation cost.

Answers to RQ4: Our results show that LS-N and LS-R perform poorly on the ISSP with the discontinuous PF. In contrast, we observed that LS-RN can address the issue of LS-N and LS-R for the discontinuous PF. Our results show that LS-RN obtains better subsets than LS-N and LS-R while reducing the number of subset evaluations and computation time. These observations indicate the importance of using two neighbor lists with different properties sequentially to handle discontinuous PFs.

D. Effects of l

Here, we investigate the effects of the candidate list size, including the nearest neighbor list size l^N and the random neighbor list size l^R . First, this section investigates the influence of l^N on the performance of LS-N. Then, this section investigates the effects of the ratio of l^N and l^R in LS-RN.

1) *Effects of l^N in LS-N:* Fig. 7 shows the results of LS-N with $l^N \in \{10, 20, \dots, 100\}$ on the ISSP with $d = 4$, $n = 5000$, and $k = 100$ using HV and ϵ . Fig. 7 shows the average number of subset evaluations required to complete the search. Fig. S.14 shows the results for all seven quality indicators. Table S.7 also shows the quality of subsets found by LS-N.

We do not describe Table S.7 in detail, but it shows that LS-N with $l^N = 10$ achieves the poorest performance in terms of the quality of subsets. We can see that LS-N obtains better subsets as l^N increases. In contrast, as shown in Fig. 7, the number of subset evaluations in LS-N increases as l^N increases. These observations indicate that l^N in LS-N controls the trade-off between the quality of subsets and computational cost (i.e., the number of subset evaluations and computation time). As seen from Table S.7, the quality of subsets is not drastically improved when setting l^N to more than 30. Although we set l^N to 40 throughout this paper, this observation shows that this setting is reasonable. Note that similar trends were observed for l^R in LS-R.

2) *Effects of the ratio of l^N and l^R in LS-RN:* This paper set the sizes of the nearest neighbor list l^N and random neighbor list l^R to 20. Here, we investigate how the ratio of l^N and l^R influences the performance of LS-RN. In this experiment, we fixed the sum of l^N and l^R at 80 in LS-RN. Then, we changed l^N as follows: $l^N \in \{0, 10, \dots, 80\}$. For example, $l^R = 70$ when $l^N = 10$. Note that LS-RN with $l^N = 0$ and $l^N = 80$ are exactly the same as LS-R and LS-N, respectively.

Table S.8 shows the quality of subsets found by LS and LS-RN with various ratios of l^N and l^R on the ISSP using the seven quality indicators. Table S.8 shows the results for the linear PF, $d = 4$, $n = 5000$, and $k = 100$. We do not provide a detailed explanation of Table S.8, but the results show that extremely small and large l^N values deteriorate the performance of LS-RN, e.g., $l^N = 0$ and $l^N = 80$.

Fig. 8 shows the results of LS-RN in terms of the average number of subset evaluations and computation time. As seen from Fig. 8, the setting of l^N does not significantly influence the two measurements of LS-RN. This observation suggests that setting l^N and l^R to the same size as in this work is a reasonable first choice on an unseen real-world ISSP.

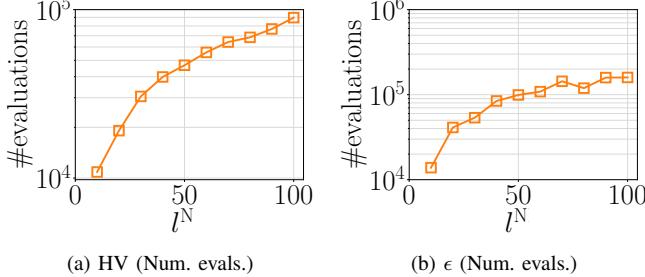


Fig. 7: Results of LS-N with $l^N \in \{10, 20, \dots, 100\}$ on the ISSP using HV and ϵ .

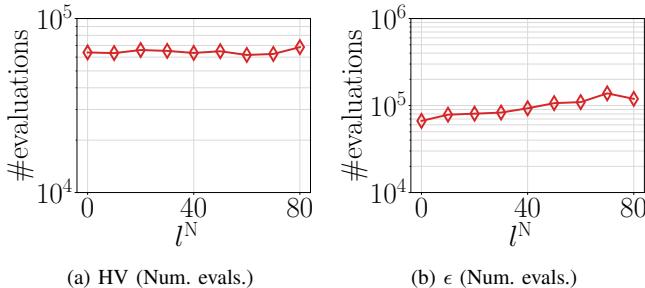


Fig. 8: Results of LS-RN with $l^N \in \{0, 10, \dots, 80\}$ and $l^R = 80 - l^N$ on the ISSP using HV and ϵ .

Answers to RQ5: Our results suggest that the nearest neighbor list size l^N in LS-N can control the trade-off between the quality of subsets and computational budget used in the search. For LS-N, the use of a large l^N value requires high computational cost but leads to improvement of the quality of subsets. We also observed that the performance of LS-RN can be improved by setting l^N and l^R to almost the same value when fixing the sum of l^N and l^R to 80.

E. Further investigations

1) *Comparison to greedy search methods:* This paper has investigated the effectiveness of LS with the candidate list strategy by comparing it to conventional LS. This section compares the proposed methods to advanced greedy search (GS) methods. Here, we consider the following four GS methods:

- GS Greedy Subset Selection [5]
- GS-L Lazy GS for HV, IGD, IGD⁺ [18]
- GS-A Approximated GS for HV [17]
- GS-AL Lazy Approximated GS for HV [17]

GS is the most basic and general method, which can be applied to the ISSP using any quality indicator. In contrast, GS-A and GS-AL can be applied only to the ISSP using HV, and GS-L was designed only for the ISSP using HV, IGD, and IGD⁺. For this reason, we performed the comparison only on the ISSP using HV. The MATLAB implementations of GS-L, GS-A, and GS-AL are publicly available. However, C++ code is generally faster than MATLAB codes. For fair comparison,

TABLE III: Comparison of the four LS methods and the four GS methods on the ISSPs using HV. The mean, standard deviation, and relative error based on GS of the quality indicator values of the subsets found by the other methods are shown.

(a) linear PF			
	mean	std.	rel. error
GS	1.373e+00	NA	
GS-L	1.373e+00	NA	
GS-A	1.372e+00	NA	3.471e-02%
GS-AL	1.372e+00	NA	3.471e-02%
LS	1.374e+00	6.142e-05	-7.976e-02%
LS-N	1.374e+00	7.190e-05	-7.522e-02%
LS-R	1.372e+00	2.111e-04	4.672e-02%
LS-RN	1.374e+00	8.525e-05	-7.213e-02%

(b) discontinuous PF			
	mean	std.	rel. error
GS	5.930e-01	NA	
GS-L	5.930e-01	NA	
GS-A	5.742e-01	NA	3.159e+00%
GS-AL	5.742e-01	NA	3.159e+00%
LS	5.947e-01	9.490e-05	-2.886e-01%
LS-N	5.889e-01	1.520e-03	6.768e-01%
LS-R	5.909e-01	5.500e-04	3.491e-01%
LS-RN	5.942e-01	1.649e-04	-2.174e-01%

we implemented all four GS methods in C++ by referring to the original MATLAB implementations. As in Section VI-D, we used the following settings: $d = 4$, $n = 5000$, $k = 100$.

Table III shows the results of the four GS methods and the four LS methods (LS, LS-N, LS-R, and LS-RN) on the ISSP using HV in terms of the quality of subsets. Tables III (a) and (b) show the results for the linear and discontinuous PFs, respectively. We performed a single run for each GS method due to its deterministic nature. For this reason, the standard deviation is not available for the four GS methods. The right columns of Tables III (a) and (b) show the relative error based on the results of GS. Since GS-L is a faster version of GS, GS-L and GS find exactly the same subset. For this reason, the relative errors for GS and GS-L are not shown in Table III. Tables S.9 and S.10 show the comparison of the two GS methods (GS and GS-L) and the four LS methods on the ISSPs using the other six quality indicators. We do not explain Tables S.9 and S.10 in detail, but they are similar to the results in Table III.

As seen from Tables III(a) and (b), LS and LS-RN find better subsets than the four GS methods. Although LS-N outperforms the four GS methods for the linear PF, LS-N is outperformed by GS and GS-L for the discontinuous PF. This poor performance of LS-N for the discontinuous PF is consistent with the observation in Section VI-C.

Figs. 9 (a) and (b) show the wall-clock time of the run on the ISSPs with linear and discontinuous PFs, respectively. As seen from Figs. 9 (a) and (b), as expected, conventional LS is the slowest in terms of the computation time. GS-L is the fastest, followed by GS-AL. Interestingly, LS-N, LS-R, and LS-RN are faster than conventional GS and slightly slower than GS-A. In summary, our results showed the effectiveness of the LS-RN compared to the four GS methods.

2) *Comparison on real-world problems:* This section investigates the performance of the proposed methods on 14

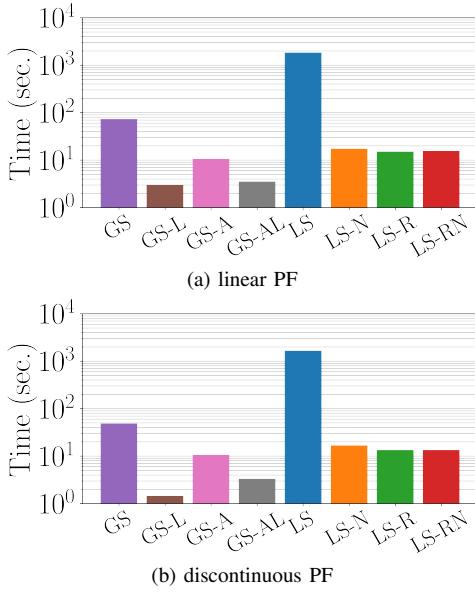


Fig. 9: Wall-clock time of the run of the four LS methods and the four GS methods on the ISSP instances using HV.

real-world problems (RE problems) [50]. Here, we consider selecting $k = 100$ points from the approximated PF P of size n of each RE problem provided by [50], where n is $500 \times d$ in the RE problems, except for the RE3-4-6 problem. Since n of the RE3-4-6 problem is only 28, we do not use it. For each RE problem, the approximated PF is normalized using the approximated ideal and nadir points.

Table IV shows the comparison of LS, LS-N, LS-R, and LS-RN on the ISSP using HV and IGD⁺ in terms of the quality of the subsets. Table S.11 shows the results for IGD, R2, NR2, ϵ , and s -energy.

As seen from Table IV(a), LS-RN is competitive to LS for HV. The relative error of LS-RN is 0.0735% even in the worst case. As shown in Table IV(b), LS-RN finds much worse subsets than LS for the RE2-3-5 and RE3-4-3 problems. On these two problems, some points in the approximated PFs are very far from other points, where this kind of approximated PF can be viewed as a special case of the discontinuous PF. In addition, such isolated points are unlikely to be included in the nearest neighbor lists of other points. These are the reasons why LS-RN performs poorly on the ISSPs based on the RE2-3-5 and RE3-4-3 problems.

The results for IGD, R2, NR2, s -energy in Table S.11 are consistent with the results in Section VI-A. That is, LS-RN is competitive to LS for most quality indicators, but LS-RN is outperformed by LS for ϵ .

Fig. 10 shows the number of subset evaluations for the four LS methods. Similar to the results in Section VI-A, we can see that the number of subset evaluations for LS-R and LS-RN is significantly lower than that for LS. In contrast, LS-N performs significantly worse than LS-R and LS-RN on the three ISSPs based on the RE2-4-3, RE2-2-4, RE3-4-2 problems in terms of the number of subset evaluations. Fig. S.18 shows the results for IGD, IGD⁺, R2, NR2, ϵ , and s -energy. Similar to Fig. 10, Fig. S.18 shows that LS-RN performs significantly better than

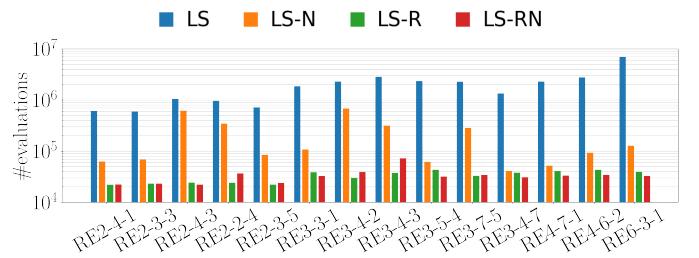


Fig. 10: Number of subset evaluations by I for the four LS methods on the ISSPs using HV and the approximated PFs of the 14 real-world (RE) problems.

LS in terms of the number of subset evaluations. In summary, the results on the ISSPs based on the RE problems show the effectiveness of LS-RN.

VII. CONCLUSION

First, this paper investigated the property of the ISSP (Section III). Our results showed that swapping two points far away from each other and close to each other is effective at early and later stages of LS, respectively. Then, based on this observation, we proposed the candidate list strategy to speed up LS for the ISSP (Section IV). This paper introduced two types of candidate lists: the nearest neighbor and random neighbor lists. This paper also presented the idea of using the two neighbor lists sequentially. We investigated the performance of the proposed candidate list strategy on the ISSPs with the seven quality indicators and various PFs (Section VI). The results for the continuous PF showed the effectiveness of the nearest neighbor list, except on the ISSP using ϵ . In contrast, our results indicated that the sequential use of the two neighbor lists is effective on the ISSP with both the continuous and discontinuous PFs. We demonstrated that the sequential use of the two neighbor lists can drastically speed up LS in terms of both the number of subset evaluations and the computation time while maintaining the quality of subsets. We found that the proposed candidate list strategy is more effective as the point set size n increases.

Our results showed that the properties of the ISSP using ϵ are different from those using other quality indicators (e.g., HV and IGD). For this reason, the proposed candidate list strategy does not work well on the ISSP using ϵ . Based on these observations, we do not recommend applying the candidate list strategy to the real-world ϵ subset selection problem. Future work should address this issue.

Our findings indicate the effectiveness of reducing the neighborhood size in LS, which has been previously overlooked. The proposed candidate list strategy can be a useful clue to design an efficient LS method for the ISSP. As described in Section I, the ISSP can be found in environmental selection in indicator-based EMO algorithms. It is interesting to conduct a systematic benchmarking of subset selection methods for the ISSP for this setting. Our results showed that the best candidate list strategy depends on various factors in the ISSP, including the distribution of points in P and the choice of quality indicator. Fitness landscape analysis of the

TABLE IV: Comparison of the four LS methods on the ISSPs using HV and the approximated PFs of the 14 real-world (RE) problems. The mean and (relative error based on LS) of the quality indicator values of the subsets found by the four methods are shown.

(a) HV					
Problem	LS	LS-N	LS-R	LS-RN	
RE2-4-1	8.85e-01	8.84e-01 - (2.77e-02%)	8.84e-01 - (2.79e-02%)	8.85e-01 - (5.91e-04%)	
RE2-3-3	7.59e-01	7.59e-01 - (3.48e-02%)	7.59e-01 - (3.52e-02%)	7.59e-01 - (1.08e-03%)	
RE2-4-3	1.16e+00	1.16e+00 - (6.74e-03%)	1.16e+00 - (1.71e-03%)	1.16e+00 - (4.88e-05%)	
RE2-2-4	1.17e+00	1.17e+00 - (3.90e-02%)	1.17e+00 - (3.32e-03%)	1.17e+00 + (- 2.72e-04%)	
RE2-3-5	1.08e+00	7.74e-01 - (2.84e+01%)	1.08e+00 \approx (5.67e-06%)	1.08e+00 - (7.35e-02%)	
RE3-3-1	1.33e+00	1.33e+00 - (8.24e-06%)	1.33e+00 - (8.72e-06%)	1.33e+00 \approx (0.00e+00%)	
RE3-4-2	1.33e+00	1.33e+00 - (5.59e-04%)	1.33e+00 - (4.80e-05%)	1.33e+00 - (1.53e-05%)	
RE3-4-3	1.31e+00	1.31e+00 - (3.04e-02%)	1.31e+00 - (1.20e-03%)	1.31e+00 - (3.00e-03%)	
RE3-5-4	1.05e+00	1.04e+00 - (1.04e-01%)	1.04e+00 - (4.40e-02%)	1.05e+00 - (9.78e-03%)	
RE3-7-5	1.31e+00	1.31e+00 - (8.42e-03%)	1.31e+00 - (3.07e-03%)	1.31e+00 - (5.66e-04%)	
RE3-4-7	8.89e-01	8.89e-01 - (8.57e-02%)	8.88e-01 - (1.45e-01%)	8.89e-01 - (2.45e-02%)	
RE4-7-1	8.65e-01	8.58e-01 - (8.04e-01%)	8.63e-01 - (3.16e-01%)	8.65e-01 - (6.62e-02%)	
RE4-6-2	8.49e-01	8.46e-01 - (3.95e-01%)	8.47e-01 - (2.05e-01%)	8.49e-01 - (4.26e-02%)	
RE6-3-1	1.50e+00	1.50e+00 - (9.82e-02%)	1.50e+00 - (6.47e-02%)	1.50e+00 - (2.01e-02%)	

(b) IGD ⁺					
Problem	LS	LS-N	LS-R	LS-RN	
RE2-4-1	2.10e-03	2.28e-03 - (8.38e+00%)	2.27e-03 - (7.75e+00%)	2.12e-03 - (6.85e-01%)	
RE2-3-3	2.07e-03	2.24e-03 - (8.04e+00%)	2.22e-03 - (7.35e+00%)	2.08e-03 - (5.24e-01%)	
RE2-4-3	2.68e-04	3.34e-04 - (2.46e+01%)	2.94e-04 - (9.66e+00%)	2.71e-04 - (1.23e+00%)	
RE2-2-4	4.11e-04	6.85e-04 - (6.68e+01%)	4.39e-04 - (6.93e+00%)	4.11e-04 \approx (5.27e-02%)	
RE2-3-5	1.12e-09	2.25e-03 - (2.01e+08%)	1.12e-06 - (1.00e+05%)	1.22e-05 - (1.09e+06%)	
RE3-3-1	7.93e-08	1.76e-07 - (1.21e+02%)	9.84e-08 - (2.40e+01%)	8.72e-08 - (9.96e+00%)	
RE3-4-2	5.57e-06	9.86e-06 - (7.72e+01%)	6.15e-06 - (1.06e+01%)	5.71e-06 - (2.60e+00%)	
RE3-4-3	4.51e-09	1.23e-05 - (2.72e+05%)	1.82e-07 - (3.93e+03%)	1.15e-06 - (2.53e+04%)	
RE3-5-4	5.92e-03	6.19e-03 - (4.68e+00%)	6.31e-03 - (6.58e+00%)	5.96e-03 - (7.04e-01%)	
RE3-7-5	3.05e-04	3.50e-04 - (1.47e+01%)	3.32e-04 - (8.70e+00%)	3.12e-04 - (2.23e+00%)	
RE3-4-7	1.10e-02	1.13e-02 - (3.01e+00%)	1.18e-02 - (7.21e+00%)	1.11e-02 - (9.36e-01%)	
RE4-7-1	2.24e-02	2.30e-02 - (2.64e+00%)	2.37e-02 - (5.90e+00%)	2.27e-02 - (1.29e+00%)	
RE4-6-2	1.15e-02	1.22e-02 - (6.13e+00%)	1.22e-02 - (6.78e+00%)	1.16e-02 - (1.39e+00%)	
RE6-3-1	8.56e-03	9.18e-03 - (7.22e+00%)	9.18e-03 - (7.20e+00%)	8.69e-03 - (1.49e+00%)	

ISSP is needed for a better understanding of the performance of subset selection methods.

REFERENCES

- [1] K. Miettinen, *Nonlinear Multiobjective Optimization*. Springer, 1998.
- [2] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, 2001.
- [3] M. Basseur, B. Derbel, A. Goëffon, and A. Liefoghe, “Experiments on greedy and local search heuristics for ddimensional hypervolume subset selection,” in *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference, Denver, CO, USA, July 20 - 24, 2016*, T. Friedrich, F. Neumann, and A. M. Sutton, Eds. ACM, 2016, pp. 541–548.
- [4] J. G. Falcón-Cardona and C. A. C. Coello, “Indicator-based multi-objective evolutionary algorithms: A comprehensive survey,” *ACM Comput. Surv.*, vol. 53, no. 2, pp. 29:1–29:35, 2020.
- [5] M. López-Ibáñez, J. D. Knowles, and M. Laumanns, “On sequential online archiving of objective vectors,” in *Evolutionary Multi-Criterion Optimization - 6th International Conference, EMO 2011, Ouro Preto, Brazil, April 5-8, 2011. Proceedings*, ser. Lecture Notes in Computer Science, R. H. C. Takahashi, K. Deb, E. F. Wanner, and S. Greco, Eds., vol. 6576. Springer, 2011, pp. 46–60.
- [6] K. Bringmann, T. Friedrich, and P. Klitzke, “Generic postprocessing via subset selection for hypervolume and epsilon-indicator,” in *Parallel Problem Solving from Nature - PPSN XIII - 13th International Conference, Ljubljana, Slovenia, September 13-17, 2014. Proceedings*, ser. Lecture Notes in Computer Science, T. Bartz-Beielstein, J. Branke, B. Filipic, and J. Smith, Eds., vol. 8672. Springer, 2014, pp. 518–527.
- [7] H. Ishibuchi, Y. Setoguchi, H. Masuda, and Y. Nojima, “How to compare many-objective algorithms under different settings of population and archive sizes,” in *IEEE Congress on Evolutionary Computation, CEC 2016, Vancouver, BC, Canada, July 24-29, 2016*. IEEE, 2016, pp. 1149–1156.
- [8] J. Bader and E. Zitzler, “Hype: An algorithm for fast hypervolume-based many-objective optimization,” *Evol. Comput.*, vol. 19, no. 1, pp. 45–76, 2011.
- [9] K. Li, “A survey of multi-objective evolutionary algorithm based on decomposition: Past and future,” *IEEE Trans. Evol. Comput.*, 2025 (in press).
- [10] K. Li, Q. Zhang, S. Kwong, M. Li, and R. Wang, “Stable matching-based selection in evolutionary multiobjective optimization,” *IEEE Trans. Evol. Comput.*, vol. 18, no. 6, pp. 909–923, 2014.
- [11] M. Wu, K. Li, S. Kwong, Y. Zhou, and Q. Zhang, “Matching-based selection with incomplete lists for decomposition multiobjective optimization,” *IEEE Trans. Evol. Comput.*, vol. 21, no. 4, pp. 554–568, 2017.
- [12] H. Ishibuchi, Y. Sakane, N. Tsukamoto, and Y. Nojima, “Selecting a small number of representative non-dominated solutions by a hypervolume-based solution selection approach,” in *FUZZ-IEEE 2009, IEEE International Conference on Fuzzy Systems, Jeju Island, Korea, 20-24 August 2009, Proceedings*. IEEE, 2009, pp. 1609–1614.
- [13] L. Bradstreet, L. Barone, and L. While, “Maximising hypervolume for selection in multi-objective evolutionary algorithms,” in *IEEE International Conference on Evolutionary Computation, CEC 2006, part of WCCI 2006, Vancouver, BC, Canada, 16-21 July 2006*. IEEE, 2006, pp. 1744–1751.
- [14] Y. Nan, K. Shang, H. Ishibuchi, and L. He, “Improving local search hypervolume subset selection in evolutionary multi-objective optimization,” in *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2021, pp. 751–757.
- [15] ———, “An improved local search method for large-scale hypervolume subset selection,” *IEEE Trans. Evol. Comput.*, vol. 27, no. 6, pp. 1690–1704, 2023.
- [16] A. P. Guerreiro, C. M. Fonseca, and L. Paquete, “Greedy hypervolume subset selection in low dimensions,” *Evol. Comput.*, vol. 24, no. 3, pp. 521–544, 2016.
- [17] K. Shang, H. Ishibuchi, and W. Chen, “Greedy approximated hypervolume subset selection for many-objective optimization,” in *GECCO '21: Genetic and Evolutionary Computation Conference, Lille, France, July 10-14, 2021*, F. Chicano and K. Krawiec, Eds. ACM, 2021, pp. 448–456.
- [18] W. Chen, H. Ishibuchi, and K. Shang, “Fast greedy subset selection from large candidate solution sets in evolutionary multiobjective op-

- timization,” *IEEE Trans. Evol. Comput.*, vol. 26, no. 4, pp. 750–764, 2022.
- [19] E. Zitzler and L. Thiele, “Multiobjective optimization using evolutionary algorithms - A comparative case study,” in *Parallel Problem Solving from Nature - PPSN V, 5th International Conference, Amsterdam, The Netherlands, September 27-30, 1998, Proceedings*, ser. Lecture Notes in Computer Science, A. E. Eiben, T. Bäck, M. Schoenauer, and H. Schwefel, Eds., vol. 1498. Springer, 1998, pp. 292–304.
- [20] L. Bradstreet, L. While, and L. Barone, “Incrementally maximising hypervolume for selection in multi-objective evolutionary algorithms,” in *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2007, 25-28 September 2007, Singapore*. IEEE, 2007, pp. 3203–3210.
- [21] M. P. Hansen and A. Jaszkiewicz, “Evaluating the quality of approximations to the non-dominated set,” Poznan University of Technology, Tech. Rep. IMM-REP-1998-7, 1998.
- [22] K. Shang, H. Ishibuchi, M. Zhang, and Y. Liu, “A new R2 indicator for better hypervolume approximation,” in *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2018, Kyoto, Japan, July 15-19, 2018*, H. E. Aguirre and K. Takadama, Eds. ACM, 2018, pp. 745–752.
- [23] H. H. Hoos and T. Stützle, *Stochastic Local Search: Foundations & Applications*. Elsevier / Morgan Kaufmann, 2004.
- [24] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, “Scalable Test Problems for Evolutionary Multi-Objective Optimization,” in *Evolutionary Multiobjective Optimization. Theoretical Advances and Applications*. Springer, 2005, pp. 105–145.
- [25] J. D. Knowles and D. Corne, “On metrics for comparing nondominated sets,” in *Proceedings of the 2002 Congress on Evolutionary Computation, CEC 2002, Honolulu, HI, USA, May 12-17, 2002*. IEEE, 2002, pp. 711–716.
- [26] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca, “Performance assessment of multiobjective optimizers: an analysis and review,” *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 117–132, 2003.
- [27] M. Li and X. Yao, “Quality evaluation of solution sets in multiobjective optimisation: A survey,” *ACM Comput. Surv.*, vol. 52, no. 2, pp. 26:1–26:38, 2019.
- [28] C. A. C. Coello and M. R. Sierra, “A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm,” in *MICAI 2004: Advances in Artificial Intelligence, Third Mexican International Conference on Artificial Intelligence, Mexico City, Mexico, April 26-30, 2004, Proceedings*, ser. Lecture Notes in Computer Science, R. Monroy, G. Arroyo-Figueroa, L. E. Sucar, and J. H. S. Azuela, Eds., vol. 2972. Springer, 2004, pp. 688–697.
- [29] H. Ishibuchi, H. Masuda, Y. Tanigaki, and Y. Nojima, “Modified distance calculation in generational distance and inverted generational distance,” in *Evolutionary Multi-Criterion Optimization*, A. Gaspar-Cunha, C. Henggeler Antunes, and C. C. Coello, Eds. Cham: Springer International Publishing, 2015, pp. 110–125.
- [30] K. Shang, H. Ishibuchi, M.-L. Zhang, and Y. Liu, “A new r2 indicator for better hypervolume approximation,” in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2018, pp. 745–752.
- [31] D. P. Hardin and E. B. Saff, “Discretizing Manifolds via Minimum Energy Points,” *Notices of the AMS*, vol. 51, no. 10, pp. 1186–1194, 2004.
- [32] R. Tanabe and H. Ishibuchi, “An analysis of quality indicators using approximated optimal distributions in a 3-d objective space,” *IEEE Trans. Evol. Comput.*, vol. 24, no. 5, pp. 853–867, 2020.
- [33] O. Schütze, X. Esquivel, A. Lara, and C. A. C. Coello, “Using the averaged hausdorff distance as a performance measure in evolutionary multiobjective optimization,” *IEEE Trans. Evol. Comput.*, vol. 16, no. 4, pp. 504–522, 2012.
- [34] J. G. Falcón-Cardona, M. T. M. Emmerich, and C. A. C. Coello, “On the construction of pareto-compliant combined indicators,” *Evol. Comput.*, vol. 30, no. 3, pp. 381–408, 2022.
- [35] J. Knowles, L. Thiele, and E. Zitzler, “A tutorial on the performance assessment of stochastic multiobjective optimizers,” ETH Zurich, Tech. Rep. TIK-214, 2006.
- [36] A. Auger, J. Bader, D. Brockhoff, and E. Zitzler, “Theory of the hypervolume indicator: optimal μ -distributions and the choice of the reference point,” in *Foundations of Genetic Algorithms, 10th ACM SIGEVO International Workshop, FOGA 2009, Orlando, Florida, USA, January 9-11, 2009, Proceedings*, I. I. Garibay, T. Jansen, R. P. Wiegand, and A. S. Wu, Eds. ACM, 2009, pp. 87–102.
- [37] S. Jiang, Y. Ong, J. Zhang, and L. Feng, “Consistencies and contradictions of performance metrics in multiobjective optimization,” *IEEE Trans. Cybern.*, vol. 44, no. 12, pp. 2391–2404, 2014.
- [38] R. Tanabe and K. Li, “Quality indicators for preference-based evolutionary multiobjective optimization using a reference point: A review and analysis,” *IEEE Trans. Evol. Comput.*, vol. 28, no. 6, pp. 1575–1589, 2024.
- [39] K. Li, K. Deb, and X. Yao, “R-metric: Evaluating the performance of preference-based evolutionary multiobjective optimization using reference points,” *IEEE Trans. Evol. Comput.*, vol. 22, no. 6, pp. 821–835, 2018.
- [40] S. Bechikh, M. Kessentini, L. B. Said, and K. Ghédira, “Chapter four - preference incorporation in evolutionary multiobjective optimization: A survey of the state-of-the-art,” *Adv. Comput.*, vol. 98, pp. 141–207, 2015.
- [41] K. Bringmann, S. Cabello, and M. T. M. Emmerich, “Maximum volume subset selection for anchored boxes,” in *33rd International Symposium on Computational Geometry, SoCG 2017, July 4-7, 2017, Brisbane, Australia*, ser. LIPIcs, B. Aronov and M. J. Katz, Eds., vol. 77. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017, pp. 22:1–22:15.
- [42] K. Bringmann, T. Friedrich, and P. Klitzke, “Two-dimensional subset selection for hypervolume and epsilon-indicator,” in *Genetic and Evolutionary Computation Conference, GECCO '14, Vancouver, BC, Canada, July 12-16, 2014*, D. V. Arnold, Ed. ACM, 2014, pp. 589–596.
- [43] K. Li, R. Chen, G. Fu, and X. Yao, “Two-archive evolutionary algorithm for constrained multiobjective optimization,” *IEEE Trans. Evol. Comput.*, vol. 23, no. 2, pp. 303–315, 2019. [Online]. Available: <https://doi.org/10.1109/TEVC.2018.2855411>
- [44] S. Li, K. Li, W. Li, and M. Yang, “Evolutionary alternating direction method of multipliers for constrained multi-objective optimization with unknown constraints,” *IEEE Trans. Evol. Comput.*, 2025 (in press).
- [45] N. Mladenović and P. Hansen, “Variable neighborhood search,” *Comput. Oper. Res.*, vol. 24, no. 11, pp. 1097–1100, 1997.
- [46] F. Biscani and D. Izzo, “A parallel global multiobjective framework for optimization: pagmo,” *J. Open Source Softw.*, vol. 5, no. 53, p. 2338, 2020.
- [47] L. While, L. Bradstreet, and L. Barone, “A fast way of calculating exact hypervolumes,” *IEEE Trans. Evol. Comput.*, vol. 16, no. 1, pp. 86–95, 2012.
- [48] J. Blank, K. Deb, Y. D. Dhebar, S. Bandaru, and H. Seada, “Generating Well-Spaced Points on a Unit Simplex for Evolutionary Many-Objective Optimization,” *IEEE Trans. Evol. Comput.*, vol. 25, no. 1, pp. 48–60, 2021.
- [49] Y. Tian, X. Xiang, X. Zhang, R. Cheng, and Y. Jin, “Sampling reference points on the pareto fronts of benchmark multi-objective optimization problems,” in *2018 IEEE Congress on Evolutionary Computation, CEC 2018, Rio de Janeiro, Brazil, July 8-13, 2018*. IEEE, 2018, pp. 1–6.
- [50] R. Tanabe and H. Ishibuchi, “An easy-to-use real-world multi-objective optimization problem suite,” *Appl. Soft Comput.*, vol. 89, p. 106078, 2020.



Keisuke Korogi is a master course student at Yokohama National University, Yokohama, Japan. He received the B.S. degree in engineering from Yokohama National University in 2023. His research interests include combinatorial optimization and multi-objective optimization.



Ryoji Tanabe is an Associate Professor at Yokohama National University, Yokohama, Japan (2019–). Previously, he was a Research Assistant Professor at Southern University of Science and Technology, China (2017–2019). He was also a Post-Doctoral Researcher at Japan Aerospace Exploration Agency, Japan (2016–2017). He received the Ph.D. degree in Science from The University of Tokyo, Tokyo, Japan, in 2016. His research interests include single- and multi-objective black-box optimization, benchmarking, and automated algorithm selection.