2023 IEEE Symposium Series on Computational Intelligence (SSCI)
Mexico City, Mexico. December 5-8, 2023

Initial Populations with a Few Heuristic Solutions Significantly Improve Evolutionary Multi-objective Combinatorial Optimization

Cheng Gong*†‡, Yang Nan*, Lie Meng Pang*, Hisao Ishibuchi*, Qingfu Zhang†‡
* Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China
†Department of Computer Science, City University of Hong Kong, Kowloon Tong, Hong Kong
‡The City University of Hong Kong Shenzhen Research Institute, Shenzhen, China
chenggong6-c@my.cityu.edu.hk, nany@mail.sustech.edu.cn, panglm@sustech.edu.cn,
hisao@sustech.edu.cn, qingfu.zhang@cityu.edu.hk

***Abstract*—Population initialization is a crucial and essential step in evolutionary multi-objective optimization (EMO) algorithms. The quality of the generated initial population can significantly affect the performance of an EMO algorithm. However, few studies have focused on designing a generalized initialization method to improve the performance of EMO algorithms in solving multi-objective combinatorial optimization (MOCO) problems. Most of the existing advanced initialization methods involve complex techniques tailored to the specific characteristics of the problems to be solved. In this paper, we propose a general and effective framework of population initialization for EMO algorithms, aiming to improve their performances in solving various MOCO problems. Our approach involves the inclusion of a few specific heuristic solutions, including extreme solutions and a center solution, into the initial population. This inclusion serves to guide the evolution of the population throughout the optimization process. Our experimental results show that initial populations with a few heuristic solutions significantly improve the performance of EMO algorithms. Algorithm behavior analysis and further study are also provided, allowing for a comprehensive understanding of the effectiveness and applicability of our proposed method.**

***Index Terms*—Population initialization, Evolutionary multi-objective optimization algorithms, Multi-objective combinatorial optimization**

I. Introduction

In real-world scenarios, many problems can be modeled as multi-objective combinatorial optimization (MOCO) problems [1], such as the multi-objective traveling salesman problem (MOTSP) [2], the multi-objective vehicle routing problem (MOVRP) [3], the multi-objective assignment Problem (MOAP) [4] and the multi-objective knapsack problem (MOKP) [5]. MOCO problems form a particular class of multi-objective optimization problems (MOPs) [6], which usually involves multiple and conflicting objectives to be optimized simultaneously.

Corresponding Authors: Hisao Ishibuchi and Qingfu Zhang.

This work was supported by National Natural Science Foundation of China (Grant No. 62250710163, 62250710682), Guangdong Provincial Key Laboratory (Grant No. 2020B121201001), the Program for Guangdong Introducing Innovative and Enterpreneurial Teams (Grant No. 2017ZT07X386), the Stable Support Plan Program of Shenzhen Natural Science Fund (Grant No. 20200925174447003), Shenzhen Science and Technology Program (Grant No. KQTD2016112514355531), the Research Grants Council of the Hong Kong Special Administrative Region, China (GRF Project No. CityU11215622), and Natural Science Foundation of China (Project No: 62276223).

Generally, a multi-objective combinatorial optimization (MOCO) problem can be defined as follow:

$$\begin{aligned} &\text{minimize} \quad F(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), .., f_m(\boldsymbol{x})) \\ &\text{subject to} \quad \boldsymbol{x} \in \mathcal{X} \end{aligned} \tag{1}$$

where $\mathcal{X}$ is a finite decision space, and $F(\boldsymbol{x})$ is an m-dimensional objective vector. The decision space $\mathcal{X}$ has a specific discrete/combinatorial structure such as permutations of n cities in the MOTSP and feasible subsets of n items in the MOKP. Usually, each individual objective conflicts with each other and no single solution can optimize all of them simultaneously. Instead, we solve an MOCO problem by obtaining Pareto optimal solutions which characterize the different trade-offs among these objectives. A solution is called a Pareto optimal solution if there exists no solution that can improve one objective without deteriorating any other objective. The set of all the Pareto optimal solutions in the objective space is called the Pareto front (PF).

Usually, it is very challenging to obtain the exact Pareto optimal solutions for an MOCO problem, since solving an MOCO problem is NP-hard even if it has one objective [7]. Moreover, the number of Pareto solutions is expected to be exponentially large as the number of objectives increases [8]. In fact, it is computationally hard to determine whether a single solution is Pareto-optimal or not for most MOCO problems [9]. All these challenges make it impractical to solve MOCO problems within a reasonable computational time by using classical mathematical methods (exact optimization). Therefore, over the past years, many efforts have been devoted to developing approximate methods such as metaheuristics approaches [10]. Among different metaheuristics approaches, evolutionary multi-objective optimization (EMO) algorithms [11]–[13] are very popular since they have shown good performances in solving MOCO problems.

EMO algorithms are population-based methods with the goal of evolving a set of solutions to approximate the Pareto set. Generally, the first step of all population-based algorithms is to generate an initial population. The role of the initial population is not trivial. Intuitively, as the starting point of evolution, a promising initial population can improve the whole optimization process and reduce some computational

978-0-7381-4408-5/23/$31.00 ©2023 IEEE

costs. However, how to generate a good initial population has long been a question in the EMO community.

Currently, there is limited research focusing on the initialization of EMO algorithms for solving MOCO problems. The most commonly-used initialization method is to randomly generate a set of solutions as the initial population [14]. The random property can help to maintain a good diversity of solutions over the decision space. In addition, the random initialization method is easy to be implemented with no computational cost. Although the random initialization method can be applied to a wide range of problems, there exists a clear issue especially for large-scale problems. That is, randomly generated solutions are usually far away from the Pareto front. It needs long computation time to find good solutions around the Pareto front by multi-objective evolution from random initial solutions.

Advanced initialization methods are needed to improve the performance of EMO algorithms when we want to solve large-scale MOCO problems efficiently. However, initialization for MOCO problems is not easy due to the following reasons. Firstly, when we use an EMO algorithm to solve an MOCO problem, its search space is determined by the encoding of decision variables. For instance, traveling salesman problems typically use permutation encoding, whereas knapsack problems employ binary encoding. This implies that a well-designed initialization method should exhibit a high level of adaptability to different encoding schemes used in various MOCO problems. Moreover, defining a suitable distance measure between solutions can be challenging, particularly for certain encoding schemes such as permutation encoding. This makes it difficult to appropriately evaluate (and maintain) the diversity of initial solutions in the decision space. In some studies [15]–[17], researchers tried to modify the initialization step to improve the EMO algorithms for solving MOCO problems. However, most of their initialization methods involve complicated heuristics that require a number of additional computation resources. Moreover, their initialization methods are often problem-dependent, which lack the generalization ability to other problems with different encoding schemes.

In this paper, we propose a general and effective framework of population initialization for EMO algorithms to solve MOCO problems. Specifically, we first decompose a given MOCO problem into several different single-objective combinatorial optimization problems by using the weighted sum scalarization with specific weight vectors. Then, we approximately solve each single-objective combinatorial optimization problem to obtain a good heuristic solution. Finally, we construct an initial population which consists of the obtained heuristic solutions and other randomly generated solutions. Our main idea is to obtain only a few heuristic solutions by using a very low computation cost. These good heuristic initial solutions improve the other randomly generated initial solutions through crossover. We evaluate the effects of our proposed framework of population initialization on the performance of a classic EMO algorithm for solving different MOTSP test instances. Our experimental results show that the proposed framework of population initialization greatly improves the performance of the original EMO algorithm with the random initial population. Most importantly, the proposed framework of population initialization can be applied to various EMO algorithms with no limitation on the types of MOCO problems (if they have some heuristic algorithms).

This paper is organized as follows. First, Section II describes the proposed framework of population initialization used in this paper. Next, Section III presents the experimental results as well as algorithm behavior analysis. Finally, Section IV concludes the paper.

II. Proposed Framework of Population Initialization

In order to efficiently solve multi-objective combinatorial optimization problems, we propose an idea of including a small number of heuristic solutions in a randomly generated initial population. In this paper, the proposed idea is explained and examined for MOTSP instances.

In the first step, we decompose the original m-objective MOCO problem (e.g., MOTSP) into some single-objective combinatorial optimization problems by using weighted sum scalarization: $g(\boldsymbol{x}) = w_1 f_1(\boldsymbol{x}) + w_2 f_2(\boldsymbol{x}) + ... + w_m f_m(\boldsymbol{x})$, where $w_1, w_2, ..., w_m$ are the elements of a given weight vector $\boldsymbol{w}$. The number of single-objective problems (i.e., the number of weight vectors) is much smaller than the population size. For the m objective MOCO problem, we consider the following $m + 1$ weight vectors:

m Extreme Weight Vectors:
weight vector 1: $\boldsymbol{w}_1 = (1, 0, ..., 0)$,
weight vector 2: $\boldsymbol{w}_2 = (0, 1, ..., 0)$,
...
weight vector m: $\boldsymbol{w}_m = (0, 0, ..., 1)$,
One Center Weight Vector:
weight vector $m + 1$: $\boldsymbol{w}_{m+1} = (1/m, 1/m, ..., 1/m)$.

The first m weight vectors are called the extreme weight vectors where each weight vector assigns a non-zero weight only to a single objective. For example, extreme weight vector m has a weight of one only for the m-th objective while it has zero weights for all the other objectives. The last weight vector is called the center weight vector, which has the same weight for all objectives. We use these $m + 1$ weight vectors to include $m + 1$ heuristic solutions in a randomly generated initial population.

The next step is to obtain a heuristic solution by solving each single-objective problem. For MOTSP instances, the distance between two cities of a created single-objective TSP is the weighted sum of the distance between them corresponding to each objective. For example, the distance between two cities p and q in the single-objective problem with weight vector $\boldsymbol{w} = (w_1, w_2, ..., w_m)$ can be formulated as:

$$d_{\boldsymbol{w}}(p, q) = w_1 d_1(p, q) + w_2 d_2(p, q) + \ldots + w_m d_m(p, q) \quad (2)$$

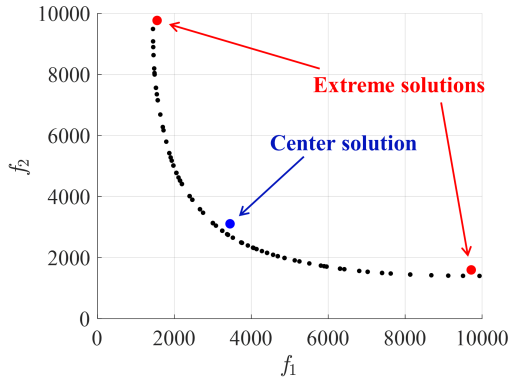


Fig. 1. Illustration of three heuristic solutions (including two extreme solutions and one center solution) obtained by solving a 2-objective TSP.

where $d_i(p,q), i = 1,2,...,m$ is the distance between two cities p and q corresponding to the i-th objective.

In this paper, we use a greedy algorithm to solve each single-objective TSP problem. Specifically, for a given single objective TSP problem, the greedy algorithm randomly picks one of the cities as the starting point. Then, it successively moves to the nearest unvisited city until produces a complete tour. Since the obtained tour depends on the choice of a starting city, we examine all cities as a starting city. Then, the tour with the shortest tour length is chosen as a heuristic solution corresponding to the given single objective TSP problem. We choose the greedy algorithm as the solver due to its small computational cost requirement for obtaining a heuristic solution. Moreover, our experimental results show that the heuristic solutions obtained by the greedy algorithm are good enough to improve the performance of EMO algorithms. Another advantage is that obtaining a single greedy solution consumes only one function evaluation, which can generally be overlooked considering the large number of generations processed by an EMO algorithm.

It is worth noting that there are several other heuristic methods available for single-objective TSP problems. Some of them can generate higher quality solutions using more computation time than the greedy algorithm. To examine the impact of the quality of heuristic solutions on our proposed framework of population initialization, we also use a state-of-the-art specialized TSP solver, the Lin-Kernighan Heuristic (LKH) solver [18], to generate heuristic solutions. We compare two heuristic algorithms in this paper: the above-mentioned greedy algorithm and the LKH solver. Using each algorithm, $m+1$ heuristic solutions are obtained. Then, the obtained heuristic solutions are included in a randomly generated initial population. Since better heuristic solutions are obtained by the LKH solver than the greedy algorithm, we can examine the effect of the quality of heuristic initial solutions on the performance of EMO algorithms.

We call each of the first m heuristic solutions (corresponding to the first m extreme weight vectors) as extreme solutions i ($i = 1,2,...,m$). The other heuristic solution (corresponding to the center weight vector $\boldsymbol{w}_{m+1}$) is called center solution $m+1$. Fig. 1 illustrates an example of three heuristic solutions obtained by solving a 2-objective TSP.

The generated $m+1$ heuristic solutions are included in a randomly generated initial population. In our computational experiments, we use the following procedure to generate an initial population of size N to compare various settings. First, an initial population of size N is randomly generated. This random initial population is used as the baseline setting. Next, k solution ($1 \leq k \leq m$) are randomly removed from the random initial population. Then, k out of the m heuristic solutions are included in the random initial population of size $N-k$ to create an initial population if size N. In this manner, we examine various initial populations (e.g., with only a single extreme solution, with all the $m+1$ heuristic solutions).

One may think that the availability of heuristic algorithms are limited for many MOCO problems (i.e., only MOTSP problems have the above-mentioned special property: A single-objective TSP problem can be created for an arbitrarily specified weight vector $\boldsymbol{w}$). In order to address this issue, we examine various settings of initial populations. For example, we examine an initial population with only a single heuristic solution. This setting corresponding the case where a heuristic algorithm is available only for one out of m objectives.

III. Experimental Studies

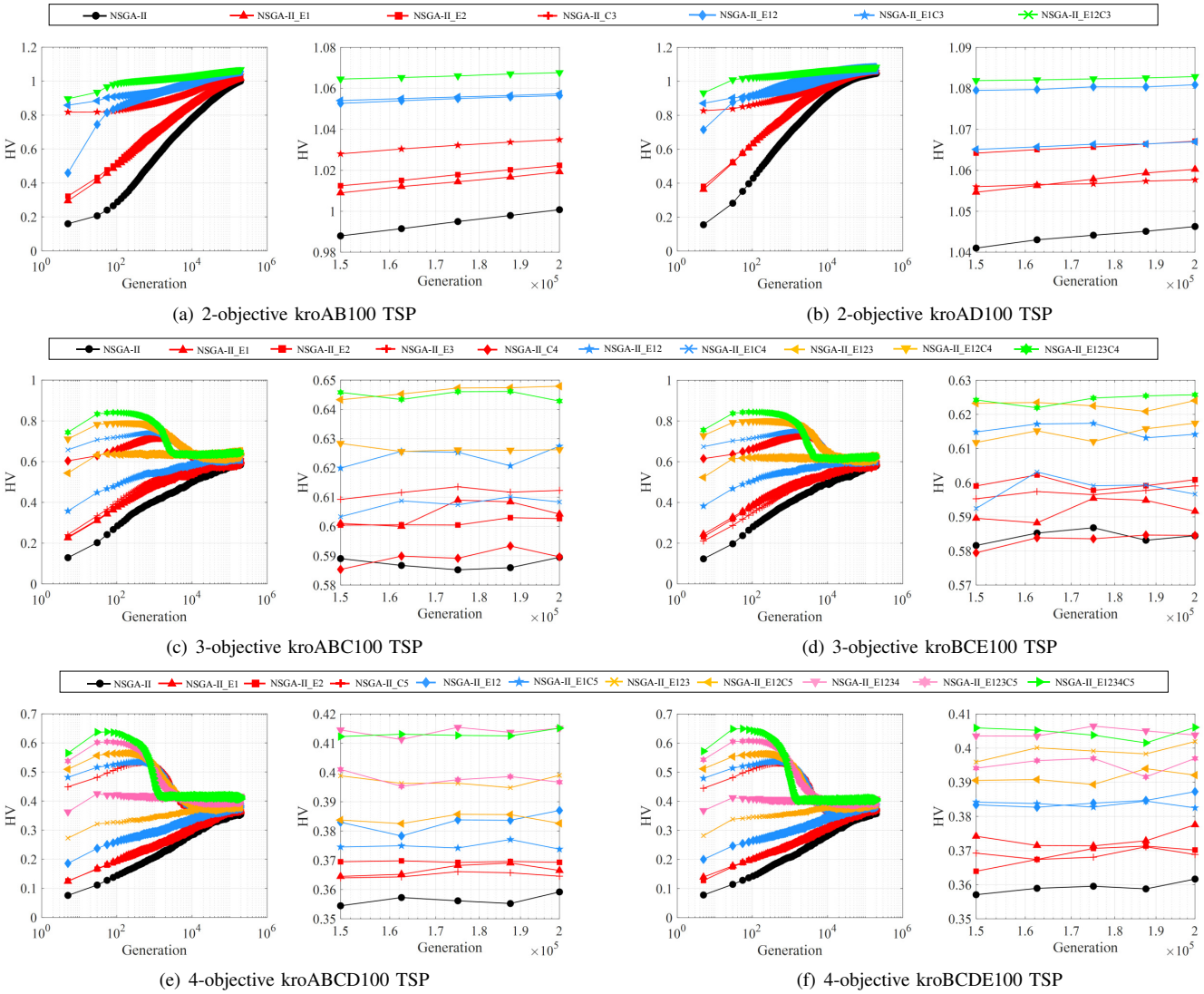
In this section, we validate the effectiveness of our proposed framework of population initialization in improving evolutionary multi-objective combinatorial optimization. Specifically, we assess its impact on the performance of an EMO algorithm compared to the original random initialization method (i.e., using randomly generated solutions as the initial population). Furthermore, we will investigate the influence of including various heuristic solutions on the evolutionary behavior of the population.

A. Experimental Setup

In this paper, we use the well-known EMO algorithm NSGA-II [11] as an example to investigate whether our proposed framework of population initialization improves its performance in solving MOCO problems. Actually, other EMO algorithms (e.g., MOEA/D [12] and NSGA-III [19]) also show similar significant performance improvement when using this proposed framework.

As explained in the previous section, $m+1$ heuristic solutions will be obtained for an m-objective TSP problem. In our experiments, we generate multiple variants of NSGA-II. Each variant of NSGA-II is named using the included heuristic solutions. For example, NSGA-II_E12 means the inclusion of extreme solutions 1 and 2 in an initial population. Similarly, NSGA-II_E1C3 means the inclusion of extreme solution 1 and center solution 3. This naming system helps us to easily understand the characteristic of each NSGA-II variant.

We evaluate the performance of these different variants as well as the original NSGA-II with a pure random initial population on the multi-objective traveling salesman problem



(a) 2-objective kroAB100 TSP

(b) 2-objective kroAD100 TSP

(c) 3-objective kroABC100 TSP

(d) 3-objective kroBCE100 TSP

(e) 4-objective kroABCD100 TSP

(f) 4-objective kroBCDE100 TSP

Fig. 2. Average anytime performance curves of the hypervolume over 31 runs for each compared algorithm on six MOTSP test instances.

TABLE I
INFORMATION ABOUT THE TEST INSTANCES

Test instances	m	D	Combined TSPLIB instances
kroAB100	2	100	kroA100 and kroB100
kroAD100	2	100	kroA100 and kroD100
kroABC100	3	100	kroA100, kroB100, kroC100
kroBCE100	3	100	kroB100, kroC100 and kroE100
kroABCD100	4	100	kroA100, kroB100, kroC100 and kroD100
kroBCDE100	4	100	kroB100, kroC100, kroD100 and kroE100

(MOTSP). In our experiment, six MOTSP test instances (with 2, 3 and 4 objectives) are generated by combining different single objective TSP instances available in TSPLIB [20]. Detailed information about the generated MOTSP test instances is shown in Table I. Specially, m denotes the number of objectives and D denotes the number of cities (i.e., the number of decision variables) in Table I.

The hypervolume metric [21] is used to compare the performances of different algorithms. For the MOTSP, which is a minimization problem, the reference point is calculated as follows [22]:

$$ref = F^{max} + 0.1 \times (F^{max} - F^{min}) \quad (3)$$

where $F^{max} = (f_1^{max}, f_2^{max}, ..., f_m^{max})$ and $F^{min} = (f_1^{min}, f_2^{min}, ..., f_m^{min})$ are respectively the maximum and minimum objective values ever found by all compared algorithms in our computational experiments.

The hypervolume of each solution set (i.e., final population) is normalized by dividing it by the hypervolume of F^{min} (which can be viewed as an estimated ideal point).

B. Experimental Results

To evaluate the anytime performance of each NSGA-II variant, we calculate and record the hypervolume of the current population at every five generations. In all algorithms on all test instances (with 2-4 objectives), the population size is set as

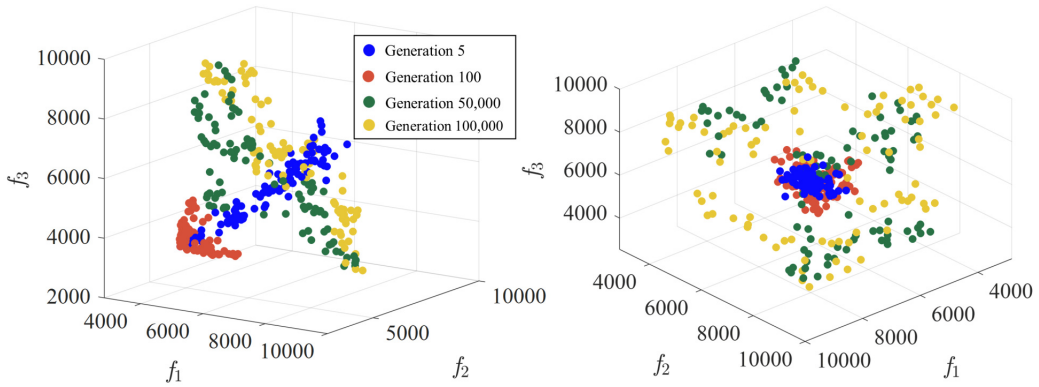


Fig. 3. Distribution of population in the objective space at specific generations for two view directions.

91. Each algorithm is terminated after $91 \times 200,000$ solutions are examined (i.e., 200,000 generations). All algorithms are executed 31 times on each test instance.

In Fig. 2, we plot the average anytime performance curves of the average hypervolume (HV) over 31 runs for each compared algorithm on each test instance. It is evident that across all 2, 3, and 4-objective TSP problems, the performance of all NSGA-II variants consistently outperforms the original NSGA-II (black points in each figure) at every generation on all the six MOTSP test instances. The results suggest that our proposed framework of population initialization, i.e., initial populations with a few heuristic solutions, significantly improves the performance of the original EMO algorithm.

We also observe that various NSGA-II variants show clearly different performances in Fig. 2. The general observations is summarized as follows:

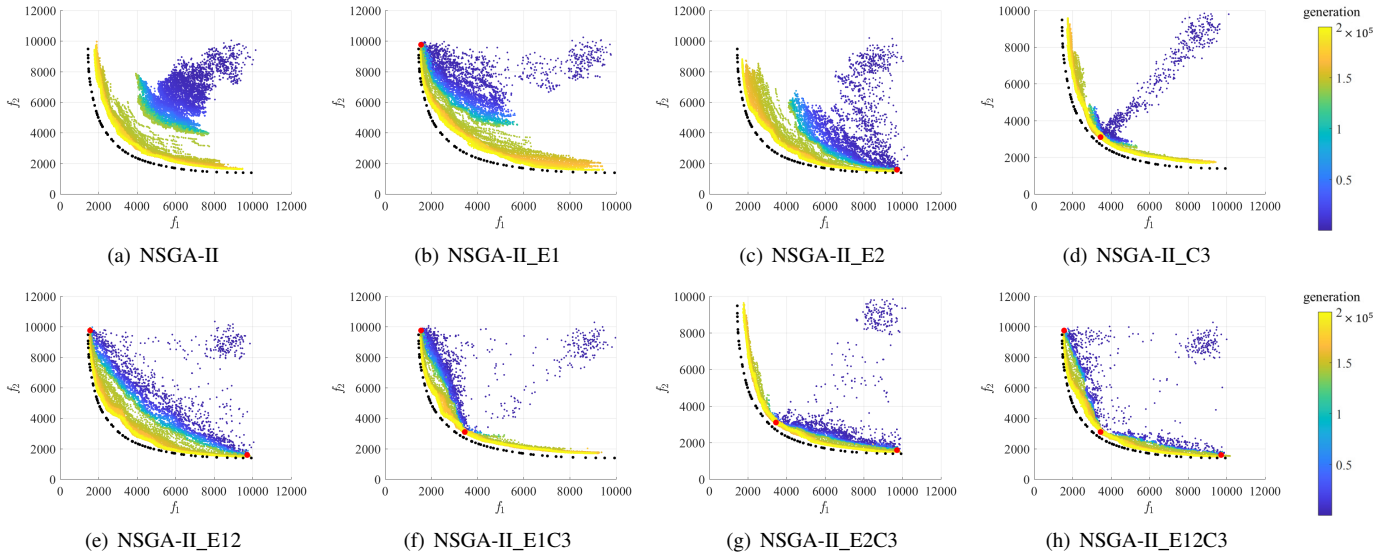
- Inclusion of any subset of the $m+1$ heuristic solutions leads to a significant performance improvement compared to the original NSGA-II with a purely random initial population.
- In general, inclusion of more heuristic solutions leads to better performance. For example, Fig. 2 (a) shows that NSGA-II_E12C3 is better than NSGA-II_E12, and NSGA-II_E12 is better than NSGA-II_E1.
- The center solution has larger effects than any extreme solution on the performance improvement especially in early generations This observation will be further examined later in the algorithm behavior analysis section.
- Among different combinations of m heuristic solutions, the best final results are obtained from the combination of all m extreme solutions. In Fig. 2 (b)-(f), the combination of all m extreme solutions shows the best performance together with the combination of all $m+1$ heuristic solutions after the 2×10^5 generations (whereas
- Among all variants, the best results in early generations are always obtained in all figures in Fig. 2 by NSGA-II with all the $m+1$ heuristic solutions. As we have just explained, NSGA-II with the m extreme solutions also show similar performance after enough generations.

In Fig. 2 (c)-(f), we observe that, for the 3- and 4-objective TSP test instances, initialization with the center solution (e.g., NSGA-II_C4, NSGA-II_E1C4, and NSGA-II_E123C4 in Fig. 2 (c)) leads to the decrease of the average hypervolume values after about 100 generations. To explain this phenomenon, we select NSGA-II_C4 in Fig. 2 (c) on kroABC100 as an example. Fig. 3 shows the distribution of its population in the objective space at some specific generations. We can observe that the population initially converges towards the PF and concentrate around the center of the PF (as shown by the solutions at the $100th$ generation: red points in Fig. 3). This behavior explains the initial increase of the average hypervolume value in Fig. 2 (c), which reaches its maximum value around the $100th$ generation. Then, NSGA-II starts to increase the diversity of solutions. As a consequence, some solutions are pushed away from the PF, which results in the decrease in the average hypervolume value.

C. Algorithm Behavior Analysis

To demonstrate the impact of our proposed framework of population initialization on the evolutionary behavior of the population, we conduct an algorithm behavior analysis in this section.

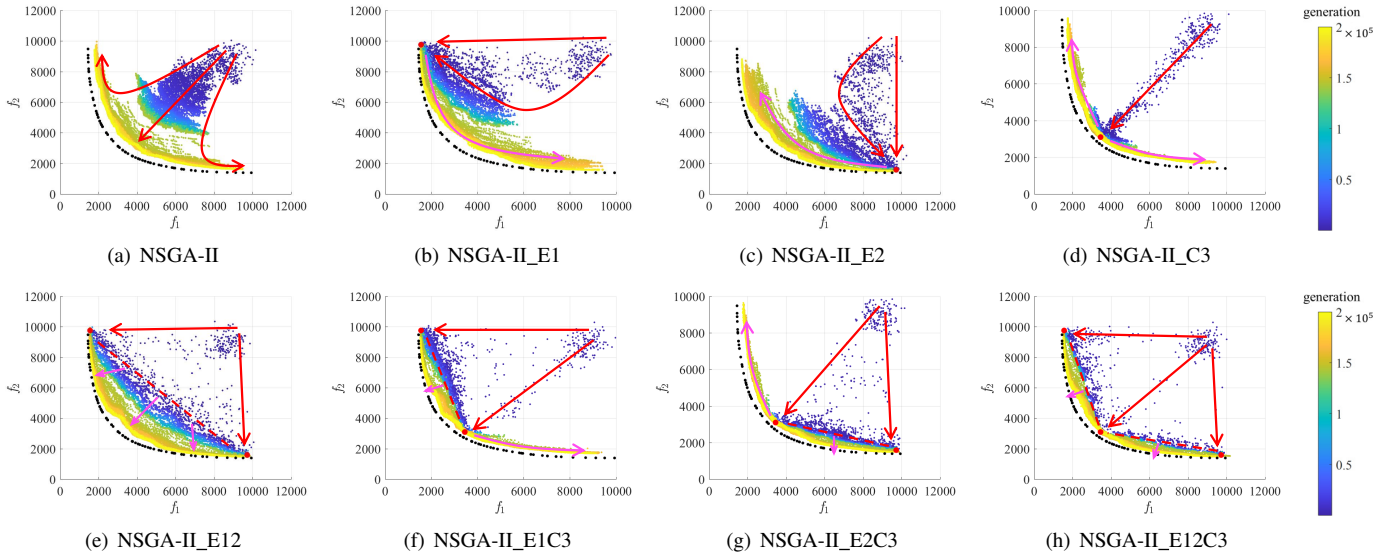
We select a single run to visualize the evolution of the population for each compared algorithm when they solve the 2-objective kroAB100 TSP test instance. As shown in Fig. 4, we plot the distribution of population in the objective space at some specific generations (i.e., 1, 3, 5, ..., 999, 1000, 2000, 3000, ..., $200000th$ generation). In each subfigure, the black points constitute the approximate Pareto front (PF) and the red points represent the included heuristic solutions. In order to obtain the approximate PF of an m-objective TSP, we first generate a large number of weight vectors $\boldsymbol{w} = (w_1, w_2, ..., w_m)$ in the same manner as in MOEA/D [12], where $w_1 + w_2 + ... + w_m = 1$ and $0 \leq w_k \leq 1 (k = 1, 2, ..., m)$. Then, for each weight vector, we formulate a single-objective problem by using the weighted sum scalarization. We use the Gurobi Optimizer [23] to solve each formulated single-objective optimization problem. Finally, the PF of each multi-objective TSP is approximated by the obtained optimal solutions [24].



(a) NSGA-II (b) NSGA-II_E1 (c) NSGA-II_E2 (d) NSGA-II_C3

(e) NSGA-II_E12 (f) NSGA-II_E1C3 (g) NSGA-II_E2C3 (h) NSGA-II_E12C3

Fig. 4. Algorithm behavior analysis of each compared algorithm for a 2-objective TSP test instance. The red points denote the included heuristic solutions and the black points denote the approximate Pareto front.



(a) NSGA-II (b) NSGA-II_E1 (c) NSGA-II_E2 (d) NSGA-II_C3

(e) NSGA-II_E12 (f) NSGA-II_E1C3 (g) NSGA-II_E2C3 (h) NSGA-II_E12C3

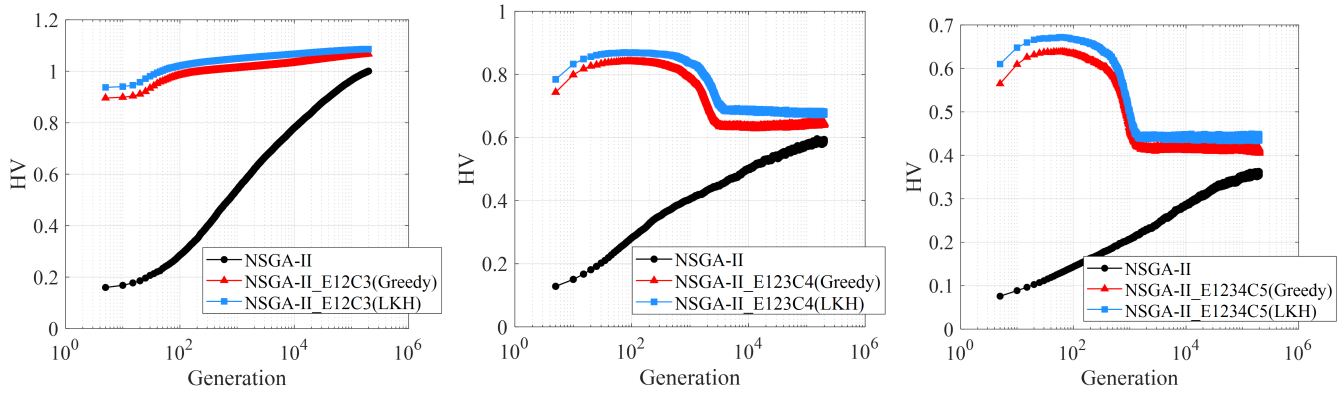
Fig. 5. Illustration of search behavior of each algorithm for a 2-objective kroAB100 TSP test instance. The red and pink arrows are used to explain the main evolutionary behavior of the population.

From Fig. 4, it can be observed that different NSGA-II variants with different initial heuristic solutions show different search behaviors. In Fig. 5, detailed illustrations of the search behavior of the population for each compared algorithm are given, aiming to provide a comprehensive understanding. We summarize the different search behaviors of the populations as follows:

(1) Fig. 5 (a) shows the search behavior of the population when a pure randomly generated initial population is used. In this case, the population will gradually converge towards the PF and simultaneously increase its diversity. We also observe that, after the middle stage of generations (e.g., as shown by the green points), the population begins to exhibit a distribution that resembles the shape of the PF.

(2) Fig. 5 (b) and Fig. 5 (c) show the search behaviors of the populations when an extreme solution is included in the initial population. In this case, first, the population will converge towards the region around the included extreme solution, as indicated by the red arrows. We can observe that the solutions farther away from the included extreme solution have a weaker convergence towards the PF. Second, the population increases its diversity by spreading the solutions across the entire PF, as indicated by the pink arrows. Also, the population continues to converge towards the PF simultaneously.

(3) Fig. 5 (d) shows the search behavior of the population when a center solution is included in the initial population. In



(a) 2-objective kroAB100 TSP (b) 3-objective kroABC100 TSP (c) 4-objective kroABCD100 TSP

Fig. 6. Comparison of the utilization of the heuristic solutions obtained by the greedy algorithm and LKH solver.

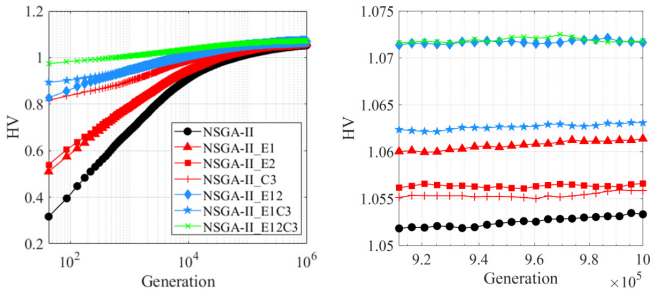


Fig. 7. Average anytime performance curves of the hypervolume over 31 runs for each compared algorithm on 2-objective kroAB100 TSP. The termination condition is set to 1,000,000 generations.

this case, first, the population will quickly converge towards the PF and concentrate around the included center solution, as indicated by the red arrow. Second, the population increases its diversity by spreading the solutions from the concentrated region towards the two sides of the PF, as indicated by the pink arrows. Also, the population slightly converges towards the PF simultaneously. As described in section 4.2, the inclusion of the center solution leads to a significant hypervolume improvement during the early generations. This phenomenon can be explained by the fact that, in the early stage of generations, a majority of the solutions concentrate around the center part of the PF, as shown by Fig. 5 (d). This will result in a large value in hypervolume calculation.

(4) Fig. 5 (e)-(h) show the search behaviors of the populations when multiple heuristic solutions are included in the initial population. In this case, first, the population will quickly converge towards an approximate hyperplane (it is essentially a line in the 2-dimensional space as shown by the red dashed line), which is formed by the region between two included heuristic solutions. Second, the population gradually converge towards the PF from the hyperplane. Additionally, when not all the extreme solutions are included (e.g., Fig. 5 (f) and (g)), the population also increases its diversity by spreading the solutions across the entire PF, as indicated by the pink arrows.

D. Further Study

In the previous section, we demonstrated that initial populations with a few heuristic solutions significantly improve evolutionary multi-objective combinatorial optimization. In this section, we take a further step to address two specific questions: (1) Does the utilization of higher-quality heuristic solutions result in greater performance improvement? (2) Does the proposed framework of population initialization consistently outperform the original random initialization approach in all generations?

To address these questions, we first use the LKH solver to generate better heuristic solutions than those generated by the greedy algorithm. Then, we compare the performances of NSGA-II variants between the two types of heuristic initial solutions: One is generated by the LKH solver and the other is generated by the greedy algorithm. In Fig. 6, we present the results for the scenario where all $m+1$ heuristic solutions are included in the initial population (i.e., the NSGA-II variant showing the best performance). From Fig. 6, we observe that using the heuristic solutions generated by LKH consistently leads to slightly better results than the case of the greedy algorithm. We also obtained similar comparison results to Fig. 6 from all the other NSGA-II variants. These results suggest that the utilization of higher-quality heuristic solutions can result in greater performance improvement. However, the magnitude of improvement is not significant (i.e., the red lines are only slightly better than the blue lines), while the computational resource consumption of LKH is higher than that of the greedy algorithm. Hence, there exists a trade-off between generating high-quality heuristic solutions and the consumption of computational resources that need to be taken into account.

Next, we extend our investigation by modifying the termination condition from 200,000 to 1,000,000. Our goal is to determine whether our proposed framework of population initialization would continue to consistently lead to performance improvement. Fig. 7 shows the average HV performance of all

compared algorithms up to 1,000,000 generations. We observe that all the NSGA-II variants using our proposed framework of population initialization continue to consistently lead to performance improvement compared to the original NSGA-II with a random initial population, even when extending the evaluation to 1,000,000 generations.

IV. Conclusion

In the paper, we proposed a general and effective framework of population initialization to improve the evolutionary multi-objective combinatorial optimization. Our main idea involves two steps for constructing the initial population: (1) Obtain a few heuristic solutions by solving the decomposed single-objective problems. Specifically, we propose obtaining m extreme solutions and one center solution for an m-objective problem; (2) Use the $m + 1$ heuristic solutions along with other randomly generated solutions to construct the initial population. We applied the proposed framework of population initialization to a well-known EMO algorithm (NSGA-II) by including some of the $m + 1$ heuristic solutions in the a randomly generated initial population. We examined the effect of heuristic initial solutions on the performance of NSGA-II. Six MOTSP test instances were used to evaluate the performance of compared algorithms. The experimental results showed that our proposed framework of population initialization can result in significant performance improvement.

Algorithm behavior analysis was provided to illustrate how heuristic initial solutions change the search behavior of NSGA-II. Our further investigation revealed two findings. One is that the utilization of higher-quality heuristic solutions can result in greater performance improvement. The other finding is that NSGA-II variants with some heuristic initial solutions is always better (independent of the choice of heuristic initial solutions from the $m+1$ heuristic solutions) than the original NSGA-II with a pure random initial population.

In our future research, we will examine our proposed framework of population initialization on other representative EMO algorithms. Additionally, we will examine the use of the proposed framework of population initialization for other MOCO problems. This will allow us to assess the generality of our proposed framework of population initialization in tackling various MOCO problems.

References

[1] E. L. Ulungu and J. Teghem, "Multi-objective combinatorial optimization problems: A survey," *Journal of Multi-Criteria Decision Analysis*, vol. 3, no. 2, pp. 83–104, 1994.

[2] C. C. Ribeiro, P. Hansen, P. C. Borges, and M. P. Hansen, "A study of global convexity for a multiple objective travelling salesman problem," *Essays and Surveys in Metaheuristics*, pp. 129–150, 2002.

[3] N. Jozefowiez, F. Semet, and E.-G. Talbi, "Multi-objective vehicle routing problems," *European Journal of Operational Research*, vol. 189, no. 2, pp. 293–309, 2008.

[4] L. Belhoul, L. Galand, and D. Vanderpooten, "An efficient procedure for finding best compromise solutions to the multi-objective assignment problem," *Computers & Operations Research*, vol. 49, pp. 97–106, 2014.

[5] H. Ishibuchi, N. Akedo, and Y. Nojima, "Behavior of multiobjective evolutionary algorithms on many-objective knapsack problems," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 2, pp. 264–283, 2014.

[6] K. Miettinen, *Nonlinear multiobjective optimization*. Springer Science & Business Media, 2012, vol. 12.

[7] M. Ehrgott and X. Gandibleux, "A survey and annotated bibliography of multiobjective combinatorial optimization," *OR-spektrum*, vol. 22, pp. 425–460, 2000.

[8] A. Herzel, S. Ruzika, and C. Thielen, "Approximation methods for multiobjective optimization problems: A survey," *INFORMS Journal on Computing*, vol. 33, no. 4, pp. 1284–1299, 2021.

[9] M. Ehrgott, *Multicriteria optimization*. Springer Science & Business Media, 2005, vol. 491.

[10] M. Ehrgott, "Approximation algorithms for combinatorial multicriteria optimization problems," *International Transactions in Operational Research*, vol. 7, no. 1, pp. 5–31, 2000.

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

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

[13] N. Beume, B. Naujoks, and M. Emmerich, "SMS-EMOA: Multiobjective selection based on dominated hypervolume," *European Journal of Operational Research*, vol. 181, no. 3, pp. 1653–1669, 2007.

[14] B. Kazimipour, X. Li, and A. K. Qin, "Effects of population initialization on differential evolution for large scale optimization," in *2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2014, pp. 2404–2411.

[15] S. Wang, X. Wang, J. Yu, S. Ma, and M. Liu, "Bi-objective identical parallel machine scheduling to minimize total energy consumption and makespan," *Journal of Cleaner Production*, vol. 193, pp. 424–440, 2018.

[16] M. Rashidnejad, S. Ebrahimnejad, and J. Safari, "A bi-objective model of preventive maintenance planning in distributed systems considering vehicle routing problem," *Computers & Industrial Engineering*, vol. 120, pp. 360–381, 2018.

[17] G. Vilcot and J.-C. Billaut, "A tabu search and a genetic algorithm for solving a bicriteria general job shop scheduling problem," *European Journal of Operational Research*, vol. 190, no. 2, pp. 398–411, 2008.

[18] K. Helsgaun, "An effective implementation of the Lin–Kernighan traveling salesman heuristic," *European Journal of Operational Research*, vol. 126, no. 1, pp. 106–130, 2000.

[19] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, 2013.

[20] G. Reinelt, "TSPLIB—a traveling salesman problem library," *ORSA Journal on Computing*, vol. 3, no. 4, pp. 376–384, 1991.

[21] J. D. Knowles, L. Thiele, and E. Zitzler, "A tutorial on the performance assessment of stochastic multiobjective optimizers," *TIK-report*, vol. 214, 2006.

[22] J. Shi, Q. Zhang, and J. Sun, "PPLS/D: Parallel Pareto local search based on decomposition," *IEEE Transactions on Cybernetics*, vol. 50, no. 3, pp. 1060–1071, 2018.

[23] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2023. [Online]. Available: https://www.gurobi.com

[24] H. Ishibuchi, L. He, and K. Shang, "Regular Pareto front shape is not realistic," in *2019 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2019, pp. 2034–2041.