A fast efficient local search-based algorithm for multi-objective supply chain configuration problem

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 28, 2020, accepted March 16, 2020, date of publication March 26, 2020, date of current version April 15, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2983473

A Fast Efficient Local Search-Based Algorithm for Multi-Objective Supply Chain Configuration Problem

XIN ZHANG[1], (Student Member, IEEE), ZHI-HUI ZHAN[1], (Senior Member, IEEE), AND JUN ZHANG[2], (Fellow, IEEE)

[1]School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China
[2]Hanyang University, Ansan 15588, South Korea

Corresponding author: Zhi-Hui Zhan (zhanapollo@163.com)

This work was supported in part by the National Key Research and Development Program of China under Grant 2019YFB2102102, in part by the Outstanding Youth Science Foundation under Grant 61822602, in part by the National Natural Science Foundations of China (NSFC) under Grant 61772207 and Grant 61873097, in part by the Guangdong Natural Science Foundation Research Team under Grant 2018B030312003, and in part by the Guangdong-Hong Kong Joint Innovation Platform under Grant 2018B050502006.

ABSTRACT Supply chain configuration (SCC) plays an important role in supply chain management. This paper focuses on a multi-objective SCC (MOSCC) problem for minimizing both the cost of goods sold and the lead time simultaneously. Some existing population-based methods use the evolution of a population to obtain the optimal Pareto set, but they are time-consuming. In this paper, an ***E***fficient ***L***ocal ***S***earch-based algorithm with ***rank*** (ELSrank) is designed to solve the MOSCC problem. Firstly, instead of use of population, two solutions ($\boldsymbol{x_A}$ and $\boldsymbol{x_B}$) are generated by the greedy strategy, which have the minimal cost and the minimal time, respectively. They approximately locate in two sides of the Pareto front (***PF***). Secondly, with the consideration of the problem characteristics, a local search (LS) is proposed to find competitive solutions among the common neighborhood of two given solutions. If $\boldsymbol{x_A}$ and $\boldsymbol{x_B}$ are chosen to execute the proposed LS, solutions along the link path (the approximate ***PF***) of $\boldsymbol{x_A}$ and $\boldsymbol{x_B}$ can be found. This way, the solutions along the whole ***PF*** can be found. The comparative experiments are conducted on six instances from the real-life MOSCC problems, and the results show that ELSrank performs better than other start-of-the-art algorithms, especially on the large scale problem instances.

INDEX TERMS Supply chain configuration, multi-objective optimization, Pareto front, local search.

I. INTRODUCTION

Supply chain configuration (SCC) is a problem to configure the supply chain network and to make choices (e.g., choosing the locations or production lines) for each supply chain member (SCM) [1]–[5]. In the problem, each SCM has several configurations, and these configurations are corresponding to different costs and times, such as the production cost/time, the assembly cost/time, and the delivery cost/time. As both the cost and time are significant for the SCC, a multi-objective SCC (MOSCC) problem is modeled and solved in this paper to optimize the total cost of goods sold (*totalCost*) and the total lead time (*totalTime*) simultaneously. The aim of the MOSCC problem is to obtain the optimal choices of SCMs and to minimize *totalCost* and *totalTime* [6], [7]. Therefore, this problem is a discrete combinatorial optimization problem (COP).

Due to the good performance of heuristic methods on the COPs, some of them have been applied to solve the MOSCC problem, especially the swarm intelligence algorithms (SIAs) [8]–[13]. For example, an ant colony optimization algorithm with Pareto optimality (P-ACO) was proposed to solve the MOSCC problem, which designed the new heuristic strategy and pheromone update rule with considering the characteristics of the problem [8]. The intelligent water drop algorithm combined with Pareto optimality (P-IWD), inspired by the flowing water drop, was also used for this problem [9]. P-ACO and P-IWD maintain a large population (i.e., the population size is 10000) to obtain a new Pareto set (***PS***) in every iteration. However, the population is much larger

The associate editor coordinating the review of this manuscript and approving it for publication was Xiangtao Li.

than the size of ***PS***. The evolution of some solutions may be not helpful to the improvement of the population. Therefore, the above two SIAs are time-consuming to maintain a large population for this problem.

In this paper, a non-population based algorithm – a novel local search is proposed to solve the MOSCC problem. This algorithm aims to find the final Pareto front (***PF***) directly and efficiently. At the beginning, two greedy solutions ($\boldsymbol{x_A}$ and $\boldsymbol{x_B}$) are generated with the minimal *totalCost* and *totalTime*, respectively, whose fitness values can be regarded as the two ends of the ***PF***. The next step is to obtain the solutions along the path (i.e., the ***PF***) between $\boldsymbol{x_A}$ and $\boldsymbol{x_B}$, and these solutions are also the common neighbors of the two solutions. From this aspect, the ***E***fficient ***L***ocal ***S***earch-based algorithm with ***rank*** (ELSrank) is proposed. Firstly, a new metric is designed to rank the configurations of SCMs. Then, the proposed local search (LS) is used to find the neighbors of two solutions by choosing the configurations whose ranks are between the ranks of the two solutions. LS is used twice in every iteration. For the first time, the two solutions used in LS include a randomly generated solution and a randomly selected solution from ***PS***, which can increase the diversity of solutions. In the second time, the two solutions are both randomly chosen from ***PS*** for generating more promising solutions to enhance exploitation.

To evaluate the performance of ELSrank on the MOSCC problem, six instances with different problem scales are used. The experimental results show that ELSrank can outperform P-ACO, P-IWD, and two famous multi-objective algorithms NSGA-II and MOEA/D, and the results of our proposed algorithm are much closer to those obtained by the enumeration method.

The rest of this paper is organized as follows. Section 2 introduces the problem model of MOSCC. Section 3 presents the details of the proposed method ELSrank. Section 4 presents and discusses the experimental results. Finally, Section 5 draws a conclusion.

II. PROBLEM MODEL OF MOSCC

The supply chain of the MOSCC problem solved in this paper is to produce and assemble products. There are three kinds of SCMs in this problem, including suppliers, assemblers, and deliverers. Some assemblers are to assemble the final products, and they are denoted as ''products'' to distinguish among different assemblers. The supply relationship of all SCMs is shown in Figure 1. From up to down, suppliers supply assemblers, and assemblers supply other assemblers and products, and products supply deliverers. It should be noted that an assembler cannot supply itself to avoid the loop, but it can supply other assemblers. Each SCM has several configurations (choices) with different costs and times, and it can choose only one. More precisely, different configurations of suppliers have different production costs and times; different configurations of assemblers and products have different assembly costs and times; different configurations of deliverers have different transportation costs and times.

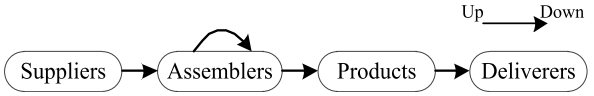


FIGURE 1. Supply relationship of SCMs.

The aim of this problem is to choose the optimal configurations of SCMs to minimize the *totalCost* and *totalTime* of the system. The decision vector $\boldsymbol{x}$ is a set of *nNode* integers which represent the choices of the corresponding SCMs, as shown in Figure 2. For example, node 1 selects the third choice, and node 2 selects the first choice. *nNode* is the number of SCMs. All up nodes of node i who supply node i are denoted as $\boldsymbol{upNode}_i$, and all down nodes of node i who node i supplies are denoted as $\boldsymbol{downNode}_i$.

	Node_id	1	2	3	...	*nNode*
$\boldsymbol{x}$	Choice	3	1	2	...	2

FIGURE 2. Decision vector $\boldsymbol{x}$ of the MOSCC problem.

The mathematical model of the problem is formulated as follows:

$$\text{Min } totalCost = nPeriod \times \sum_{i=1}^{nNode} demand_i \times Cost_{i,x_i} \quad (1)$$

$$\text{Min } totalTime = \max_{i=1,2,\ldots nNode} \{LT_i\} \quad (2)$$

$$demand_i = \sum_{j \in downNode_i} demand_j, \quad i = 1, 2, \ldots, nNode \quad (3)$$

$$LT_i = Time_{i,x_i} + \max_{j \in upNode_i} \{LT_j\}, \quad i = 1, 2, \ldots, nNode \quad (4)$$

$$x_i \in \{1, 2, \ldots, Choice_i\}, \quad i = 1, 2, \ldots, nNode \quad (5)$$

Equation (1) is the first objective function of this problem to calculate the minimal cost (*totalCost*) of the supply chain system. *nPeriod* is the number of the production period. $demand_i$ is the demand quantity of node i, which is calculated in (3), and the demand quantities of all nodes at the lowest level (deliverers) are known. $Cost_{i,x_i}$ is the production cost of the member node i on choice x_i. x_i is the decision variable, and its domain is shown in (5) where $Choice_i$ is the number of choices of node i. Equation (2) is the second objective function to calculate the minimal time (*totalTime*), where the lead time of node i (LT_i) is calculated in (4). $Time_{i,x_i}$ is the production time of node i on choice x_i.

In total, the two objectives of the MOSCC problem are *totalCost* and *totalTime*, and the decision variables are the chosen choice x_i of node i ($i = 1, 2, \ldots, Choice_i$).

III. THE PROPOSED METHOD ELSrank

In this section, the motivation of ELSrank to solve the MOSCC problem is firstly described, which is inspired by

overcoming the disadvantages of the SIAs for the problem. Then, two components of the proposed method are presented, including the generation of two greedy solutions and the proposed LS. Lastly, the complete ELSrank is described.

A. MOTIVATION OF ELSRANK FOR THE MOSCC PROBLEM

To obtain the optimal solution set of a multi-objective problem (i.e., the PS), SIAs are widely used due to their good diversity of solutions [14]–[20]. In SIAs, PS is updated by the evolution of the whole population. However, not all individuals in the population can promote the evolution, and it may cost much time to reach the real ***PF*** (***PF****). Figure 3 shows how SIAs get the final ***PF***. Firstly, they get the initial ***PF*** (i.e., the ***PF***′) which is obtained from the initial population. Then, ***PF***′ will approach to the ***PF**** continuously iteration by iteration via the improvement of the whole population in SIAs.

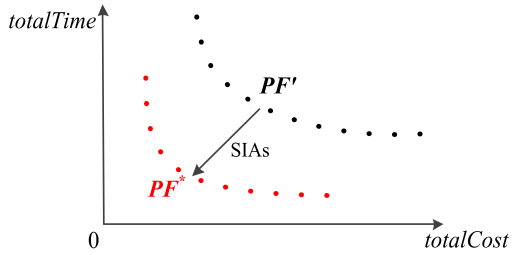


FIGURE 3. Illustration of SIAs for the MOSCC problem.

In this paper, ELSrank is proposed to obtain the ***PF*** directly by the search of the initial ***PF*** (i.e., the ***PF***″). The illustration of ELSrank is shown in Figure 4. The two ends of ***PF***″ (A and B) are obtained by the greedy strategies firstly, and they are close to the two ends of the ***PF**** (A* and B*). Then, the proposed LS is used to find the solutions along the ***PF***″, and to push ***PF***″ towards ***PF****. The detailed descriptions of Figure 4 and ELSrank are shown in the following sections.

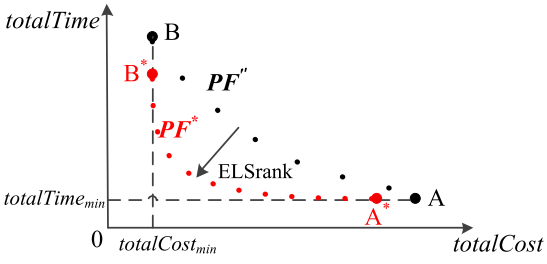


FIGURE 4. Illustration of ELSrank for the MOSCC problem.

B. GENERATION OF TWO GREEDY SOLUTIONS

Before describing the greedy solution generation, the symbols in Figure 4 are introduced. Points A* and B* are the two ends of ***PF****, and the corresponding solutions $\boldsymbol{x}_{A^*}$ and $\boldsymbol{x}_{B^*}$ have the minimal *totalTime* ($totalTime_{min}$) and *totalCost* ($totalCost_{min}$), respectively. The two greedy solutions are denoted as $\boldsymbol{x}_A$ and $\boldsymbol{x}_B$, and their locations in the figure (points A and B, respectively) can be considered as the two ends of ***PF***″.

In $\boldsymbol{x}_A$, all nodes select the choice with the minimal time, and it can be inferred from (2) and (4) that the *totalTime* of $\boldsymbol{x}_A$ is minimal and equal to that of $\boldsymbol{x}_{A^*}$ ($totalTime_{min}$). Similarly, in $\boldsymbol{x}_B$, all nodes select the choice with the minimal cost, and the *totalCost* of $\boldsymbol{x}_B$ is minimal and equal to that of $\boldsymbol{x}_{B^*}$ ($totalCost_{min}$), concluded from (1). It should be noted that the *totalCost* of $\boldsymbol{x}_A$ is not smaller than that of $\boldsymbol{x}_{A^*}$, and the *totalTime* of $\boldsymbol{x}_B$ is not smaller than that of $\boldsymbol{x}_{B^*}$.

C. THE PROPOSED LS

The classical LS is usually performed on one solution to find better solutions among its neighborhood. Besides, the neighborhood is commonly obtained by random changes of the solution, without consideration of the problem characteristics [21]–[26]. In this paper, an efficient LS is proposed to obtain the common neighborhood of two solutions (***xs*** and ***xt***). Moreover, the LS is designed with the consideration of the problem characteristics. Herein, given two solutions ***xs*** and ***xt***, their neighborhood can also be regarded as the search space along the link path of ***xs*** and ***xt***.

To find the solutions between ***xs*** and ***xt*** (along the link path), all choices of a node are ranked. For one node (i.e., a dimension of a solution), it is assumed that the corresponding choices of ***xs*** and ***xt*** are *cs* and *ct*, respectively, and the ranks of *cs* and *ct* are *rs* and *rt*, respectively. If $rs < rt$ or $rs > rt$, the solutions generated by LS will randomly choose the configurations ranking between *rs* and *rt* (i.e., [*rs*, *rt*) or [*rt*, *rs*)); otherwise, the solutions will choose the same one as ***xs*** and ***xt***. This process will repeat until a solution can be added into ***PS*** or the number of the loop exceeds the predefined parameter *nLS*.

Algorithm 1 shows the procedure of the proposed LS. Firstly, it is judged whether ***xs*** is the same as ***xt*** in lines 1 and 2. If the two solutions are the same, the procedure will terminate. Then, a loop is executed to generate

Algorithm 1 LS(***xs***, ***xt***)

```
Begin
1  If xs is same as xt
2      return;
3  num = 0;
4  While num < nLS
5      For i = 1 : nNode // generate a new solution x
6          rs ← rank of the choice xs_i of node i;
7          rt ← rank of the choice xt_i of node i;
8          x_i ← a random choice ranking between rs and
           rt;
           //i.e., a random value in [rs, rt) or [rt, rs)
9      Evaluate x and update PS and PF;
10     If x is added into PS
11         Break;
12     num++;
End
```

new solutions which are the neighbors of the ***xs*** and ***xt***. The ranks of each node of the new solutions are all between the corresponding nodes of ***xs*** and ***xt***. After evaluating the new solution ***x*** and updating ***PS*** and ***PF***, if ***x*** is a Pareto solution (i.e., it has been added into ***PS***), the loop ends; otherwise, the loop will continue and will be executed at most *nLS* times.

Figure 5 shows an example of the proposed LS. In the figure, node 1 has 4 choices, and from the first rank to the fourth rank, they are choice 4, choice 1, choice 2, and choice 3, respectively. For node 1, the choices of ***xs*** and ***xt*** are 2 and 4, whose ranks are third and first, respectively. Then, the generated solutions will select the configurations whose rank is between [1, 3). Therefore, the rank of the option can be first or second, and the corresponding choice is 4 or 1. In the example, the choices of $x1$ and $x2$ on node 1 are 1 (ranking second) and 4 (ranking first), respectively.

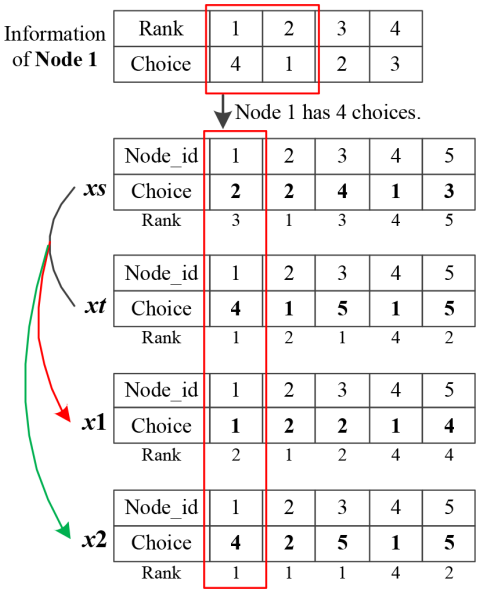


FIGURE 5. Example of the proposed LS.

Because the proposed LS is based on the rank of configurations, the design of the ranking is crucial for the algorithm. In this problem, different configurations have different costs and times. The main idea of designing the rank is to select a choice as the base, and to transform other configurations and sort them under this same base.

Firstly, for a node, the base choice is randomly chosen from the choices whose cost and time are both larger than zero. The cost and time of the base choice are denoted as *costb* and *timeb*, respectively. The base choice spends *timeb*/*costb* with the unit cost. Then, for the comparison with each other, the cost of other choices is transformed to be the spent time under the base choice. The transformation method is designed as shown in (6), where i and j are the indices of node and choice, respectively, and $\Delta time_{i,j}$ represents the difference of the transformed time and the original time.

$$\Delta time_{i,j} = [timeb - (Cost_{i,j} - costb) \times \frac{timeb}{costb}] - Time_{i,j} \quad (6)$$

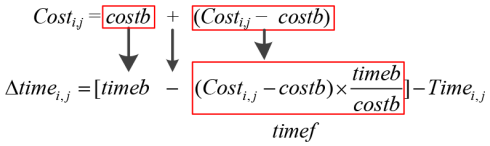


FIGURE 6. Illustration of the transformation of the cost.

In (6), $Cost_{i,j}$ is divided into two parts *costb* and ($Cost_{i,j}$–*costb*), as shown in Figure 6. *costb* is corresponding to *timeb*, and only the difference ($Cost_{i,j}$–*costb*) is transformed proportionately (($Cost_{i,j}$–*costb*)×(*timeb*/*costb*)), denoted as *timef*. As the higher cost is corresponding to the less time, the transformed time is *timeb* minus *timef*. If $\Delta time_{i,j} > 0$, it represents that with the same cost, the choice r spends less time than the base choice, and is more cost-effective. Therefore, the choice with larger $\Delta time_{i,j}$ ranks higher.

Table 1 shows an example of a node which has four choices, and the choice 2 in **bold** is selected as the base choice. In the example, the choice 1 is a little cheaper (3.25 units) than the base choice 2, but it spends twice as long as the base choice (40 units of the choice 1 compared with 20 units of the base choice). The choice 1 is less cost-effective than the base choice, and therefore, the $\Delta time$ of the choice 1 is much smaller than zero, as shown in Table 1. For the choice 4, it costs only 3.34 units more than the base choice, and it does not spend any time (having products in stock). The choice 4 is more cost-effective than the base choice, and therefore, the $\Delta time$ of the choice 4 is much bigger than zero. After sorting all choices by $\Delta time$ in descending order, their ranks are obtained and shown in the last column of Table 1.

TABLE 1. Example of the choice of a node.

Choice	*Time*	*Cost*	$\Delta time$ in (6)	Rank
1	40.00	130.00	-60.49	4
2	**20.00**	**133.25**	0.00	3
3	10.00	134.91	9.75	2
4	0.00	136.59	19.50	1

D. THE COMPLETE ELSRANK

The complete flow chart of ELSrank is shown in Figure 7. Firstly, configurations of all nodes are ranked, which will be used in the proposed LS. Two greedy solutions ($\boldsymbol{x_A}$ and $\boldsymbol{x_B}$) are generated to define two ends of ***PF***. Then, a loop is executed until the terminal condition is satisfied. During the loop, the proposed LS (LS(***xs***, ***xt***)) is used twice. In the first time, ***xs*** is randomly generated (***x***), and ***xt*** is randomly chosen from ***PS*** ($\boldsymbol{PS_r}$ where r is randomly chosen from [1, |***PS***|], and |***PS***| is the size of ***PS***). It can help the algorithm to explore the wider space and enhance the diversity. In the second time, ***xs*** and ***xt*** are both chosen from PS randomly ($\boldsymbol{PS_{ri}}$ and $\boldsymbol{PS_{rj}}$ where ri and rj are randomly chosen from [1, |***PS***|]). It is

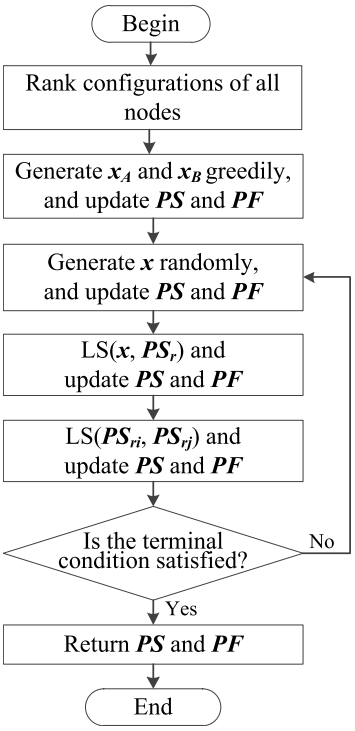


FIGURE 7. Flowchart of ELSrank.

aimed to find more promising solutions along the link path of ***xs*** and ***xt***.

IV. RESULTS AND DISCUSSIONS

This section firstly describes the experiment settings including the test instances, the evaluation criteria, and the parameter settings of comparative algorithms. Then, the experimental results of ELSrank and other algorithms are shown and discussed. Lastly, the parameter *nLS* is investigated.

A. EXPERIMENTAL SETTINGS

To evaluate the performance of our proposed algorithm, six test instances in [9] are used, as shown in Table 2. The number of solutions is equal to $\prod Choice_i$ $(i = 1, 2, \ldots, nNode)$, and the maximum execution time is the terminal condition of algorithms. Two metrics are used in this paper, including hypervolume (HV) and C(A, B) (A and B represent two algorithms) [27]–[31]. Besides, four competitive algorithms are compared with ELSrank, including P-ACO [8], P-IWD [9], NSGA-II [32], and MOEA/D [33], and the exhaustive enumeration is also used. Since the enumeration is time-consuming on instances 5 and 6, it is not compared on the two instances. Each algorithm is run 30 times independently on each instance.

B. EXPERIMENTAL RESULTS

Four competitive algorithms are used for the comparisons. P-ACO and P-IWD were proposed to solve the same MOSCC problem [8], [9]. NSGA-II and MOEA/D are classical and excellent to solve the multi-objective problem [32], [33]. The parameters are the same as the literature.

TABLE 2. Descriptions of six test instances.

Instance	Number of solutions	Maximum execution time (seconds)
1	3.07×10^3	1.0
2	6.14×10^3	1.0
3	2.45×10^4	1.0
4	4.19×10^6	30.0
5	2.74×10^{11}	100.0
6	1.28×10^{16}	500.0

Table 3 shows the comparisons of algorithms on HV which is transformed into a ratio value. HV is calculated on each run, and the mean value of all runs are recorded. If the ratio value is bigger than zero, it represents that ELSrank performs better than the corresponding algorithm. As shown in Table 3, the ratio values of P-ACO, P-IWD, NSGA-II, and MOEA/D are all bigger than zero, and it shows that ELSrank can get better results than them.

TABLE 3. Comparisons of ELSrank with compared algorithms on HV (HV(ELSrank)/HV(other algorithm) - 1).

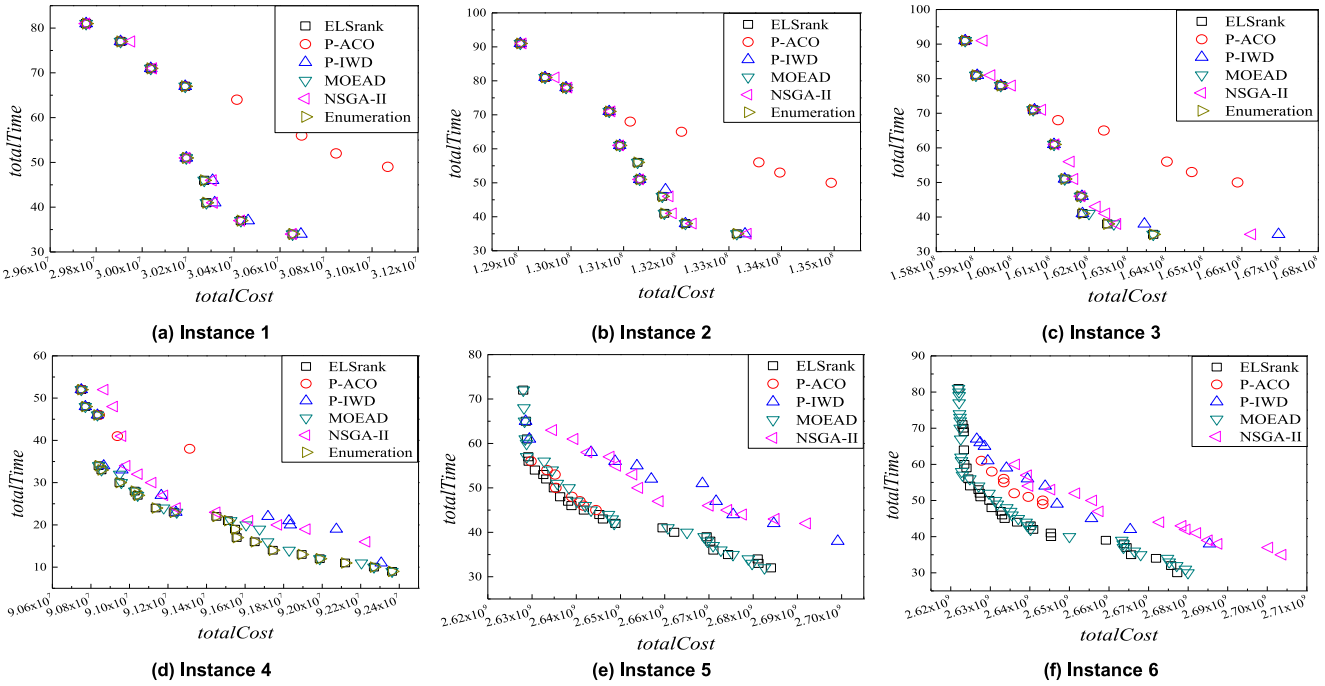
Instance	P-ACO	P-IWD	MOEA/D	NSGA-II
1	55.88%	40.46%	2.48%	10.84%
2	46.63%	39.90%	1.96%	16.14%
3	48.32%	49.61%	9.09%	17.98%
4	159.63%	66.93%	11.24%	53.15%
5	28.06%	107.17%	11.10%	97.60%
6	54.79%	67.42%	4.86%	76.43%
Average	65.55%	61.92%	6.79%	45.36%

Table 4 shows the comparative results on C(A, B). For an algorithm, all Pareto solutions of 30 run times are recorded as a whole to calculate the C(A, B). If C(A, B) is higher than C(B, A), it represents that the solutions of the algorithm B are dominated by or equal to those of the algorithm A more. It can be seen from Table 4 that C(ELSrank, -) is bigger than C(-, ELSrank) on all compared algorithms. It concludes that the number of dominations of solutions of ELSrank over solutions of other algorithms is more than that of other algorithms over ELSrank. Moreover, most of C(ELSrank, -) is equal to 1.00, which represents that all solutions in the compared algorithms are dominated by those in ELSrank.

Figure 8 shows the ***PF*** of all algorithms. In Figure 8 (a), (b), and (c), the ***PF*** of MOEA/D is close to that of ELSrank, and both of them are close the true ***PF*** generated by the enumeration method. With the problem scale increasing, in Figure 8 (d) which represents the results on instance 4, the ***PF*** of ELSrank is still close to the true ***PF***. However, the performance of MOEA/D begins to deteriorate. In Figure 8 (e) and (f), ELSrank gets the ***PF*** which is nearest to the coordinate axes.

TABLE 4. Comparisons of ELSrank with compared algorithms on C(A, B).

Instance	P-ACO		P-IWD		MOEAD		NSGA-II	
	C(ELSrank, -)	C(-, ELSrank)	C(ELSrank, -)	C(-, ELSrank)	C(ELSrank, -)	C(-, ELSrank)	C(ELSrank, -)	C(-, ELSrank)
1	**1.00**	0.44	**1.00**	0.56	**1.00**	1.00	**1.00**	0.44
2	**1.00**	0.36	**1.00**	0.64	**1.00**	1.00	**1.00**	0.18
3	**1.00**	0.40	**1.00**	0.50	**1.00**	0.80	**1.00**	0.00
4	**1.00**	0.05	**1.00**	0.14	**1.00**	0.48	**1.00**	0.00
5	**1.00**	0.00	**0.91**	0.04	**0.54**	0.40	**1.00**	0.00
6	**1.00**	0.00	**1.00**	0.00	**0.53**	0.38	**1.00**	0.00
Average	**1.00**	0.21	**0.98**	0.31	**0.84**	0.68	**1.00**	0.10



(a) Instance 1 (b) Instance 2 (c) Instance 3

(d) Instance 4 (e) Instance 5 (f) Instance 6

FIGURE 8. PF of all algorithms. (a) instance 1 (b) instance 2 (c) instance 3 (d) instance 4 (e) instance 5 (f) instance 6.

TABLE 5. Comparisons of ELSrank (nLS = 5) with different nLS on C(A, B).

Instance	$nLS = 10$		$nLS = 15$		$nLS = 20$		$nLS = 25$	
	C(ELSrank, -)	C(-, ELSrank)	C(ELSrank, -)	C(-, ELSrank)	C(ELSrank, -)	C(-, ELSrank)	C(ELSrank, -)	C(-, ELSrank)
1	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
3	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
4	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
5	**0.68**	0.40	**0.63**	0.32	**0.93**	0.08	**0.88**	0.16
6	**0.58**	0.38	**0.77**	0.24	**0.90**	0.07	**0.86**	0.14
Average	**0.88**	0.80	**0.90**	0.76	**0.97**	0.69	**0.96**	0.72

In conclusion, ELSrank is outperformed than other algorithms, followed by MOEA/D. The reason may be that ELSrank does not rely on a population and gets the ***PF*** directly, which helps to save more time to improve the ***PF***.

C. PARAMETER STUDY

There is only a parameter *nLS* needed to be studied. *nLS* is the maximum execution number of the proposed LS. Since the terminal condition of the algorithm is the given execution time, if *nLS* is large, it will cost a lot of time to execute the LS; otherwise, if *nLS* is small, the chance to improve solutions will decrease. Therefore, the setting of *nLS* is significant to the proposed algorithm.

Five different values of *nLS* are tested, including 5, 10, 15, 20, and 25. The C(A, B) of them is shown in Table 5, and the default value of *nLS* in ELSrank is 5. From the observation of Table 5, it can be seen that C(ELSrank, -) and C(-, ELSrank) are both equal to 1.00 on instances 1 to 4. It represents that the results of them are all the same on small scale problems. For instances 5 and 6 with the larger problem scales, ELSrank with $nLS = 5$ performs better than other settings of *nLS*. Therefore, *nLS* is set to 5 in the ELSrank algorithm.

V. CONCLUSION

In this paper, the ELSrank algorithm is proposed to solve the MOSCC problem for minimizing the *totalCost* and *totalTime*. Instead of using the evolution of a population to find promising solutions, ELSrank firstly obtains two greedy solutions as two ends of the ***PF***. Then, the rank of configurations is designed for the proposed LS to explore the solution space along the link path of two solutions. The proposed LS is also aimed to depict and improve the whole PF of the algorithm. The experiments show that our proposed algorithm has a better performance than the compared algorithms.

REFERENCES

[1] Z. Lu, K. Deb, E. Goodman, and J. Wassick, "Solving a supply-chain management problem using a bilevel approach," in *Proc. Genetic Evol. Comput. Conf. (GECCO)*, Berlin, Germany, 2017, pp. 1185–1192.

[2] F. Ye, Z. Xie, Z. Cai, and Q. Lin, "Optimization of the biofuel supply chain with capital-constrained farmers under government subsidies," *IEEE Access*, vol. 8, pp. 8178–8192, 2020.

[3] X. Zhang, K.-J. Du, Z.-H. Zhan, S. Kwong, T.-L. Gu, and J. Zhang, "Cooperative coevolutionary bare-bones particle swarm optimization with function independent decomposition for large-scale supply chain network design with uncertainties," *IEEE Trans. Cybern.*, early access, Sep. 20, 2019, doi: 10.1109/tcyb.2019.2937565.

[4] G. Soni, V. Jain, F. T. S. Chan, B. Niu, and S. Prakash, "Swarm intelligence approaches in supply chain management: Potentials, challenges and future research directions," *Supply Chain Manage., Int. J.*, vol. 24, no. 1, pp. 107–123, Jan. 2019.

[5] A. A. Taleizadeh, S. T. A. Niaki, and A. Makui, "Multiproduct multiple-buyer single-vendor supply chain problem with stochastic demand, variable lead-time, and multi-chance constraint," *Expert Syst. Appl.*, vol. 39, no. 5, pp. 5338–5348, Apr. 2012.

[6] S. C. Graves and S. P. Willems, "Optimizing the supply chain configuration for new products," *Manage. Sci.*, vol. 51, no. 8, pp. 1165–1180, Aug. 2005.

[7] S. H. Amin and G. Zhang, "An integrated model for closed-loop supply chain configuration and supplier selection: Multi-objective approach," *Expert Syst. Appl.*, vol. 39, no. 8, pp. 6782–6791, Jun. 2012.

[8] L. A. Moncayo-Martínez and G. Recio, "Bi-criterion optimisation for configuring an assembly supply chain using Pareto ant colony meta-heuristic," *J. Manuf. Syst.*, vol. 33, no. 1, pp. 188–195, Jan. 2014.

[9] L. A. Moncayo-Martínez and E. Mastrocinque, "A multi-objective intelligent water drop algorithm to minimise cost of goods sold and time to market in logistics networks," *Expert Syst. Appl.*, vol. 64, pp. 455–466, Dec. 2016.

[10] L. A. Moncayo-Martínez and D. Z. Zhang, "Multi-objective ant colony optimisation: A meta-heuristic approach to supply chain design," *Int. J. Prod. Econ.*, vol. 131, no. 1, pp. 407–420, May 2011.

[11] B. Yuce, E. Mastrocinque, A. Lambiase, M. S. Packianather, and D. T. Pham, "A multi-objective supply chain optimisation using enhanced bees algorithm with adaptive neighbourhood search and site abandonment strategy," *Swarm Evol. Comput.*, vol. 18, pp. 71–82, Oct. 2014.

[12] S. Zhang, C. K. M. Lee, K. Wu, and K. L. Choy, "Multi-objective optimization for sustainable supply chain network design considering multiple distribution channels," *Expert Syst. Appl.*, vol. 65, pp. 87–99, Dec. 2016.

[13] B. L. Shankar, S. Basavarajappa, J. C. H. Chen, and R. S. Kadadevaramath, "Location and allocation decisions for multi-echelon supply chain network—A multi-objective evolutionary approach," *Expert Syst. Appl.*, vol. 40, no. 2, pp. 551–562, Feb. 2013.

[14] Z.-H. Zhan, J. Li, J. Cao, J. Zhang, H. S.-H. Chung, and Y.-H. Shi, "Multiple populations for multiple objectives: A coevolutionary technique for solving multiobjective optimization problems," *IEEE Trans. Cybern.*, vol. 43, no. 2, pp. 445–463, Apr. 2013.

[15] Y. Lin, Y.-S. Jiang, Y.-J. Gong, Z.-H. Zhan, and J. Zhang, "A discrete multiobjective particle swarm optimizer for automated assembly of parallel cognitive diagnosis tests," *IEEE Trans. Cybern.*, vol. 49, no. 7, pp. 2792–2805, Jul. 2019.

[16] X.-F. Liu, Z.-H. Zhan, Y. Gao, J. Zhang, S. Kwong, and J. Zhang, "Coevolutionary particle swarm optimization with bottleneck objective learning strategy for many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 23, no. 4, pp. 587–602, Aug. 2019.

[17] Z.-G. Chen, Z.-H. Zhan, Y. Lin, Y.-J. Gong, T.-L. Gu, F. Zhao, H.-Q. Yuan, X. Chen, Q. Li, and J. Zhang, "Multiobjective cloud workflow scheduling: A multiple populations ant colony system approach," *IEEE Trans. Cybern.*, vol. 49, no. 8, pp. 2912–2926, Aug. 2019.

[18] S. Mirjalili, "Dragonfly algorithm: A new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems," *Neural Comput. Appl.*, vol. 27, no. 4, pp. 1053–1073, May 2016.

[19] Q. Feng, Q. Li, P. Chen, H. Wang, Z. Xue, L. Yin, and C. Ge, "Multi-objective particle swarm optimization algorithm based on adaptive angle division," *IEEE Access*, vol. 7, pp. 87916–87930, 2019.

[20] J. Yang and J. Liu, "Influence maximization-cost minimization in social networks based on a multiobjective discrete particle swarm optimization algorithm," *IEEE Access*, vol. 6, pp. 2320–2329, 2018.

[21] X. Zhang, X. Li, and J. Wang, "Local search algorithm with path relinking for single batch-processing machine scheduling problem," *Neural Comput. Appl.*, vol. 28, no. S1, pp. 313–326, Dec. 2017.

[22] H. R. Lourenço, O. C. Martin, and T. Stützle, "Iterated local search: Framework and applications," in *Handbook of Metaheuristics*. Cham, Switzerland: Springer, 2019.

[23] H. H. Hoos, "Stochastic local search," *Handbook of Approximation Algorithms and Metaheuristics: Methologies and Traditional Applications*. Boca Raton, FL, USA: CRC Press, 2018.

[24] A. Jaszkiewicz, "Many-objective Pareto local search," *Eur. J. Oper. Res.*, vol. 271, no. 3, pp. 1001–1013, Dec. 2018.

[25] M. Etscheid and H. Röglin, "Smoothed analysis of local search for the maximum-cut problem," *ACM Trans. Algorithms*, vol. 13, no. 2, pp. 1–12, Mar. 2017.

[26] J. Ji, S. Gao, S. Wang, Y. Tang, H. Yu, and Y. Todo, "Self-adaptive gravitational search algorithm with a modified chaotic local search," *IEEE Access*, vol. 5, pp. 17881–17895, 2017.

[27] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 257–271, Nov. 1999.

[28] M. Li and X. Yao, "Quality evaluation of solution sets in multiobjective optimisation: A survey," *ACM Comput. Surveys*, vol. 52, no. 2, pp. 1–38, Mar. 2019.

[29] D. Martínez-Vega, "Evaluation of the evolutionary algorithms performance in many-objective optimization problems using quality indicators," in *Nature-Inspired Design of Hybrid Intelligent Systems*. Cham, Switzerland: Springer, 2017.

[30] T. Sağ, "Performance assessment of multi-objective optimization algorithms on large-scale problems," in *Proc. 5th Int. Conf. Innov. Sci. Technol.*, Barcelona, Spain, Dec. 2018.

[31] G. Yu, Y. Jin, and M. Olhofer, "Benchmark problems and performance indicators for search of knee points in multiobjective optimization," *IEEE Trans. Cybern.*, early access, Feb. 11, 2019, doi: 10.1109/tcyb.2019.2894664.

[32] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.
[33] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, Dec. 2007.

XIN ZHANG (Student Member, IEEE) received the B.S. and M.S. degrees from Northeast Normal University, Changchun, China, in 2014 and 2017, respectively. She is currently pursuing the Ph.D. degree with the South China University of Technology, Guangzhou, China.

Her research interests include evolutionary computation, swarm intelligence, and their applications in real-world problems, such as supply chain network design.

ZHI-HUI ZHAN (Senior Member, IEEE) received the bachelor's and Ph.D. degrees from the Department of Computer Science, Sun Yat-Sen University, Guangzhou, China, in 2007 and 2013, respectively.

From 2013 to 2015, he was a Lecturer and an Associate Professor with the Department of Computer Science, Sun Yat-sen University. Since 2016, he has been a Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, where he is currently the Changjiang Scholar Young Professor and the Pearl River Scholar Young Professor. His current research interests include evolutionary computation algorithms, swarm intelligence algorithms, and their applications in real-world problems, and in environments of cloud computing and big data.

Dr. Zhan was a recipient of the Outstanding Youth Science Foundation from the National Natural Science Foundations of China (NSFC), in 2018, and the Wu Wen Jun Artificial Intelligence Excellent Youth from the Chinese Association for Artificial Intelligence, in 2017. His doctoral dissertation was awarded the China Computer Federation (CCF) Outstanding Dissertation and the IEEE Computational Intelligence Society (CIS) Outstanding Dissertation. He is listed as one of the Most Cited Chinese Researchers in Computer Science. He is also an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and *Neurocomputing*.

JUN ZHANG (Fellow, IEEE) received the Ph.D. degree from the City University of Hong Kong, Hong Kong, in 2002.

He is currently a Visiting Scholar with Victoria University, Melbourne, VIC, Australia. His current research interests include computational intelligence, cloud computing, high performance computing, operations research, and power electronic circuits.

Dr. Zhang was a recipient of the Changjiang Chair Professor from the Ministry of Education, China, in 2013, the China National Funds for Distinguished Young Scientists from the National Natural Science Foundation of China, in 2011, and the First-Grade Award in Natural Science Research from the Ministry of Education, China, in 2009. He is also an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, the IEEE TRANSACTIONS ON CYBERNETICS, and the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONIC.