Hindawi
Security and Communication Networks
Volume 2022, Article ID 4967775, 16 pages
https://doi.org/10.1155/2022/4967775

Research Article

A Novel Self-Adaptive Mixed-Variable Multiobjective Ant Colony Optimization Algorithm in Mobile Edge Computing

Yiguang Gong,[1] **Weixue Wang**,[1] **and Siqi Gong**[2]

[1]*School of Automation, Nanjing University of Information Science & Technology, Nanjing 210000, China*
[2]*School of Water Resources and Hydropower Engineering, WuHan University, Wuhan 430000, China*

Correspondence should be addressed to Yiguang Gong; yiguang-gong@nuist.edu.cn

Received 15 May 2021; Revised 19 October 2021; Accepted 21 January 2022; Published 8 March 2022

Academic Editor: Xuyun Zhang

Mobile edge computing (MEC) provides physical resources closer to end users, becoming a good complement to cloud computing. The booming MEC brings many multiobjective optimization problems. The paper proposes a multiobjective optimization (MOO) algorithm called $SAMOACO_{MV}$, which provides a new choice for solving MOO problems of MEC. We improve the ACO_{MV} algorithm that is only suitable for solving mixed-variable single-objective optimization (SOO) problems and propose a $MOACO_{MV}$ algorithm suitable for solving mixed-variable MOO problems. And aiming at the dependence of $MOACO_{MV}$ algorithm performance on parameter setting, we proposed the $SAMOACO_{MV}$ algorithm using a self-adaptive parameter setting scheme. Furthermore, the paper also designs some mixed-variable MOO benchmark problems for the purpose to test and compare the performance of the $SAMOACO_{MV}$ algorithm. The experiments indicate that the $SAMOACO_{MV}$ algorithm has excellent comprehensive performance and is an ideal choice for solving mixed-variable MOO problems.

1. Introduction

In recent years, mobile edge computing (MEC), as a powerful computing paradigm, provides sufficient computing resources for the internet of things (IoT) [1]. Edge computing extends traditional cloud services to the edge of the network and closer to users and is suitable for network services with low latency requirements. There are many multiobjective optimization (MOO) problems in MEC, and the research on MOO for MEC is also a hot topic. Liu et al. [1] propose a multiobjective resource allocation method, named MRAM, and the method is leveraged to optimize the time cost of IoT applications, load balance, and energy consumption of MEC servers. Huang et al. [2] present a multiobjective whale optimization algorithm (MOWOA) based on time and energy consumption to solve the optimal offloading mechanism of computation offloading in MEC. Fan et al. [3] propose an algorithm based on particle swarm optimization (PSO) to solve the MOO of the container-based microservice scheduling, aiming to optimize network latency among microservices, reliability of microservice applications, and load balancing of the cluster.

Xu et al. [4] present a multiobjective computation offloading method (MOC) for internet of vehicles (IoV) in MEC to realize the multiobjective optimization of decreasing the load balancing rate and reduce the energy consumption in ECDs and shorten the time during processing the computing tasks.

This paper studies the multiobjective optimization algorithm, which provides a new choice for MOO in MEC. The classic MOO algorithm converts the multiple objective function values into a single value according to certain rules and then applies single-objective optimization algorithms to solve them [5]. There are three common converting rules [6]: weighted sum of multiple objective function values, calculating the distance between the objective function value vector and a given decision vector and finding the maximum value of the relative difference between the respective objective function values and their corresponding given values. The classic MOO algorithm is essentially a single-objective optimization algorithm, which cannot really solve the MOO

problem. Most of the modern MOO algorithms are heuristic algorithms that can find the Pareto solution set. Some famous algorithms are NSGA-II [7], SPEA2 [8], PAES [9], and NSGA-III [10] based on an evolutionary algorithm; SMPSO [11] and OMOPSO [12] based on particle swarm algorithm; GDE3 [13], MOEAD [14], and MOEA/D-IEpsilon [15] based on differential evolution algorithm; MOACO [16], P-ACO [17], MACS [18], Monaco [19], and SACO [20] based on ant colony algorithm; and so on. Other heuristic MOO algorithms include: MOO algorithms based on simulated annealing, tabu search and immune algorithms, and new algorithms obtained by improving or mixing various algorithms. According to the no free lunch (NFL) theorems in [21], when dealing with MOO problems, the average performance of various algorithms is the same, but the algorithms can show different performances for different optimization problems. Therefore, it is another hot spot for scholars to study the applicable algorithms for specific optimization problems or to study the applicable problems according to the characteristics of optimization algorithms.

Refer to the literature [22] for the classification of optimization problems, MOO problems can be divided into four categories according to whether their variable domains are continuous or not:

(i) Continuous-variable (CV) MOO: the range of all variables is the continuous domain. These continuous variables are usually mapped to real numbers

(ii) Pseudo-discrete variable (PDV) MOO: the range of all variables is ordered discrete domain, which means that the variable values are arranged in ascending or descending order according to certain rules. The pseudo-discrete variables are usually mapped to integers.

(iii) Real-discrete-variable (RDV) MOO: the range of all variables is a disordered discrete domain, which means that the variable values cannot be arranged according to certain rules. The discrete variables are usually called categorical variables.

(iv) Mixed-variable MOO: the range of the variables includes continuous domain and discrete domain.

According to the NFL theorem, in order to obtain better optimization performance, different types of MOO problems should use different types of optimization algorithms. The research on continuous-variable MOO and pseudo-discrete variable MOO is relatively mature. Most of the aforementioned heuristic algorithms or their variants are suitable for solving these two types of problems. There are a few studies on mixed-variable MOO.

Manson et al. [23] present a novel Bayesian multiobjective algorithm (MVMOO) capable of simultaneously optimizing both discrete and continuous input variables. The algorithm utilizes Gaussian processes as surrogates in combination with a novel distance metric based upon Gower similarity. MVMOO was able to perform competitively when compared to NSGA-II with a substantially reduced experimental budget, providing a viable, efficient option when optimizing expensive mixed-variable multiobjective optimization problems.

Li et al. [24] propose an improved version of OLAR-PSO-d named OLAR-PSO-DE. The OLAR-PSO-DE utilizes a modified stagnation strategy and a dynamic hybridization strategy. The OLAR-PSO-DE is employed to optimize the design of the engine hood, which is a high-dimensional, multiobjective, and mixed-variable optimization problem. The comparative study and final hood optimization results prove that the proposed method can effectively solve complicated engineering problems.

Khokhar et al. [25] modify the continuous-variable version of the PSP algorithm to handle mixed variables. The performance of PSP was tested using a set of quality indicators with a benchmark test suite. And the performance was compared with the state-of-the-art multiobjective optimization algorithms. The modified PSP is found to be competitive when the total number of function evaluations is limited but faces an increased computational challenge when the number of design variables increases.

However, there are relatively few studies on discrete variable MOO and mixed-variable MOO, but such MOO problems are often encountered in engineering. Therefore, the research on these two types of MOO algorithms is of great significance. This paper proposes the $\text{SAMOACO}_{\text{MV}}$ algorithm by improving the ACO_{MV} algorithm [26]. The main work of the author is as follows:

(i) Improve the ACO_{MV} algorithm used to solve mixed-variable SOO problems to make it suitable for solving mixed-variable MOO problems

(ii) Propose a self-adaptive parameter setting scheme for the algorithm and verify the superiority of the self-adaptive parameter setting scheme by comparison with the manual parameter adjustment scheme

(iii) Design some mixed-variable MOO benchmark problems to test and compare the performance of the $\text{SAMOACO}_{\text{MV}}$ algorithm

(iv) Apply $\text{SAMOACO}_{\text{MV}}$ algorithm to solve spring design engineering problems and compare the algorithm performance with other well-known MOO algorithms

2. Materials and Methods

2.1. ACO_{MV} Algorithm. The ACO_{MV} algorithm [26] is an ant colony optimization algorithm proposed by K. Socha and M. Dorigo for solving mixed-variable problems. The algorithm has excellent comprehensive performance when dealing with mixed-variable optimization problems, but for pure continuous optimization or pure discrete optimization, it has weaker performance than some specialized algorithms.

The basic process of the ACO_{MV} algorithm is as follows: the first step is to initialize the solution archive by randomly creating some solutions and storing them in the solution archive. In the second step, the ants construct some new solutions based on the solution archive. Many algorithms, such as local search, gradient descent, can be used to construct and improve the quality of new solutions. The

third step is to refresh the solution archive with the new solutions, and the best solutions will be stored in the solution archive. Repeat steps 2 and 3 until the termination criteria are met.

2.1.1. The Structure, Initialization, and Refresh of Solution Archive. $\mathrm{ACO_{MV}}$ maintains a solution archive T, whose dimension $|T| = k$ can be set in advance. Assume that there is an n-dimensional continuous optimization problem that has k feasible solutions, $\mathrm{ACO_{MV}}$ stores n variable values of each feasible solution and its objective function value in the solution archive. Figure 1 depicts the structure of the solution archive, where s_j^i represents the value of the i-th variable of the j-th solution and w_j represents the weight of the j-th solution. The solutions in the solution archive are sorted by their quality (such as the value of the objective function), so the position of the solution in the archive reflects its preference (pheromone).

Before the algorithm starts, k solutions are randomly generated and stored in the solution archive T. In each iteration of the algorithm, m ants generate m new solutions. The new solutions and the solutions from the solution archive T form a solution set including $k+m$ solutions and take the k solutions with the best quality (such as objective function value) from the solution set to refresh solution archive T. The solutions in the solution archive are always sorted by their quality, and the best quality solution is at the top. In this way, the search process will always tend to find the best quality solution, so as to achieve the solution of the optimization problem.

2.1.2. Constructing New Solutions Probabilistically. Each ant constructs a new solution incrementally, that is, selects the value of the solution variable one by one. First, the ants select a solution from the solution archive based on the selection probability. The selection probability of the j-th solution is as follows:

$$P_j = \frac{\omega_j}{\sum_{r=1}^{k} \omega_r}, \tag{1}$$

where ω_j can be calculated by using various formulas. In this paper, the Gaussian function $g(\mu, \sigma) = g(1, qk)$ is selected, which formula is as (2). Besides, q is the algorithm parameter, and k is the number of solutions in the solution archive.

$$\omega_j = \frac{1}{qk\sqrt{2\pi}} e^{\left(-(j-1)^2/2q^2k^2\right)}. \tag{2}$$

Then, construct a new solution based on the selected solution. According to the probability density function $P(x)$ for each dimension variable of the solution, the ant probabilistically extracts a new value in the neighborhood of the variable value of the solution, and these new values form a new solution. For different types of variables, the structure of the probability density function is different.

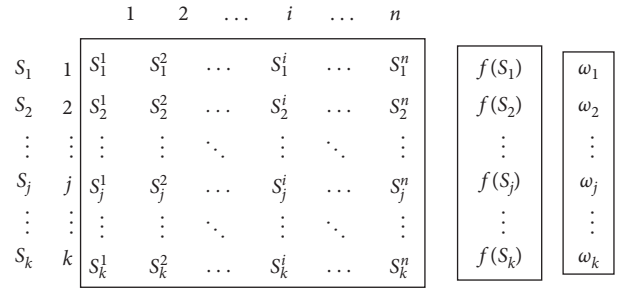

Figure 1: The structure of solution archive.

The $P(x)$ of continuous variables is as follows:

$$P(x) = g(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\left(-(x-\mu)^2/2\sigma^2\right)}, \tag{3}$$

$$\mu = s_j^i \sigma = \xi \sum_{r=1}^{k} \frac{\left|s_r^i - s_j^i\right|}{k-1}, \tag{4}$$

where $g(x, \mu, \sigma)$ represents the Gaussian function with the variable x, μ is the mean value, σ is the mean square error, and ξ is the algorithm parameter.

The $P(x)$ of ordered discrete variables is the same as (3), but it needs to be modified as follows:

(i) The variable x is the index number of the ordered discrete variable value in its range. If the variable value range of x is {large, medium, small}, then $x = 1$ when the variable value is "large", $x = 2$ when the variable value is "medium", and $x = 3$ when the variable value is "small".

(ii) The new value obtained by probability extraction according to $P(x)$ needs to be rounded to the closest value of the index number in the domain. If the extracted value is 2.3, it needs to be rounded to 2, which corresponds to "middle."

The probability density function of disordered discrete variables is as follows:

$$O_l^i = \frac{w_l}{\sum_{r=1}^{c} w_r}, \tag{5}$$

where O_l^i represents the probability of selecting the l-th variable value from the domain $D_i = \left\{v_1^i, \ldots, v_{c_i}^i\right\}$ of the i-th variable of the solution. And ω_l is the weight associated with the l-th available value; it is calculated as follows:

$$w_l = \frac{\omega_{jl}}{u_l^i} + \frac{q}{\eta}, \tag{6}$$

where ω_{jl} is the weight corresponding to the best quality solution in the solution archive whose value of the dimension variable is not empty, and it can calculate as (2). In particular, if this dimension variable of all solutions is empty, then ω_{jl} is taken as 0. u_l^i is the number of solutions whose value of this dimension variable is not empty in the solution archive. q is an algorithm parameter, which is the same as q in (2). η is the number of unused values in the domain D_i of the dimension variable.

2.2. $MOACO_{MV}$ Algorithm. Improve the single-objective optimization algorithm ACO_{MV} and obtain the $MOACO_{MV}$ algorithm suitable for solving MOO problems. The main improvement of the $MOACO_{MV}$ algorithm is to introduce the Pareto set into the solution archive, which is the non-inferior solution set [27]. The specific method is to sort according to the Pareto characteristics of the solution in the solution archive, and the solution with the best quality is placed at the top of the solution archive. After improvement, the probability of selecting a good solution is higher so that the $MOACO_{MV}$ algorithm can find non-inferior solutions.

The solutions in the solution archive are arranged according to the following two rules:

(i) The solutions in the solution archive are sorted according to the non-inferior order, and the solutions with the smaller order value are arranged at the top of the solution archive. Referring to reference [28], the definition of non-inferior order of the solution is in Definition 1.

(ii) For solutions with the same non-inferior order, they are sorted according to the degree of congestion of the solution, and the solution with a lower degree of congestion is ranked at the top of the solution archive. Referring to reference [9], the definition of the congestion degree of the solution is in Definition 2.

In the above two rules, the first rule ensures that the algorithm can find non-inferior solutions, and the second rule ensures that the distribution of these non-inferior solutions is as uniform as possible. The $MOACO_{MV}$ algorithm designed according to the above rules has excellent comprehensive performance.

Definition 1. Non-inferior order of one solution $NIO(S_j)$: In the solution set $T = \{S_1, \ldots S_j \ldots S_k\}$, take out its non-inferior solutions to form a solution set *TU(z)* whose sequence number $z = 0$, and the remaining solutions refresh the solution set T; repeat the above process until T is an empty set, and every time it is repeated, z increases by 1. Then the $NIO(S_j)$ is the sequence number z of the non-inferior solution set *TU(z)* in which the solution S_j is.

Definition 2. Congestion degree of the solution $C\ D(S_x)$: In the solution set $T = \{S_1, \ldots S_i \ldots S_k\}$, the objective function corresponding to the solution S_j is $F(S_j) = (f_1(S_j) \ldots f_i(S_j) \ldots f_v(S_j))$. Calculate the distance between $F(S_x)$ of one solution S_x and $F(S_y)$ of other solutions and take out the minimum distance $d(x, T)$. The calculation process of $d(x, T)$ is as equation (1). Then the $C\ D(S_x)$ is $d(x, T)$ multiplied by the adjustment coefficient α; the calculation process is as equation (2).

$$d(x, T) = \min_{y \in T, y \neq x} \left\| F(S_x) - F(S_y) \right\|^2, \quad CD(S_x) = \alpha \times d(x, T). \tag{7}$$

2.3. $SAMOACO_{MV}$ Algorithm. The ant colony algorithm needs to set some parameters, which have a huge impact on the performance of the algorithm. Since the convergence speed of the algorithm and the diversity of the solution are always contradictory, how to obtain a compromised excellent performance through proper parameter settings is the purpose of studying parameter settings.

This paper adopts the self-adaptive parameter control method to adjust the parameters of the $MOACO_{MV}$ algorithm according to the quality of the solution archive and the convergence speed of the algorithm. And we call this MOO algorithm as $SAMOCO_{MV}$ algorithm.

The $SAMOACO_{MV}$ algorithm needs to set four parameters, which are: the convergence speed ξ, the size of search solution archiving area q, the number of ants m, and the solution archive size k. In this paper, to balance the diversity and convergence abilities of $SAMOACO_{MV}$, two modifications for four parameters are proposed.

2.3.1. Set Method for Parameters ξ and q. The parameter ξ is used to adjust the convergence speed of the algorithm, and the parameter q is used to change the size of the search area. These two parameters are in conflict. When the search area increases or the convergence speed decreases, more Pareto solutions can be found with higher probability, but the calculation time becomes longer, and vice versa. In order to obtain a good Pareto solution archive with reasonable calculation time, we calculate the quality index of the solution archive and adjust the parameter ξ and q according to the value of the quality index. The set method for parameters ξ and q is shown in Algorithm 1:

In Algorithm 1, the quality index $P_i(T)$ of the solution archive for the i-th iteration is calculated firstly. The $P_i(T)$ is the mean value of the weighted sum of each objective function and congestion degree of all solutions in the solution archive. Next, the quality index's increment $\Delta P_i(T)$ and the parameters' increment Δe_i, Δq_i is calculated. Finally, the new parameter values are set by subtracting the product of $\Delta\xi_i$, $\Delta\ q_i$, step size constant B, and random number r from the old parameter values.

2.3.2. Set Method for Parameters m and k. The parameter m is the number of ants, and the parameter k is the solution archive size. The larger the values of these two parameters are, the higher the probability of obtaining more Pareto solutions is, but the larger parameters' value will also bring more calculations and increase time-consuming. We set the expected number of Pareto solutions according to the complexity of the problem and then adjust these two parameters in real time according to the difference between *ENUM* and the actual number of Pareto solutions. *ENUM* is the number of non-inferior solutions expected from the solution archive. The set method for parameters m and k is shown in Algorithm 2.

In Algorithm 2, count the number of solutions whose non-inferior order $NIO_i(S_j)$ are zero in the solution archive. Then calculate the ratio factors *rateArchive* and *rateAnt*, which represent the size of the solution archive and the

Input: $F_i(S_j)$, $CD_i(S_j)$, ξ_i, q_i, where $i \in (1, l-1)$, $j \in (1, k)$;
(1) $P_i(T) = (1/k)\sum_{j\in(1,k)}(\beta \cdot F_i(S_j) + \gamma CD_i(S_j))$
(2) $\Delta P_i(T) = P_i(T) - P_{i-1}(T)$
(3) $\Delta\xi_i = \xi_i - \xi_{i-1}$
(4) $\Delta q_i = q_i - q_{-1}$
(5) $\xi_{i+1} = \xi_i - r * B * \Delta P_i(T) * \Delta\xi_i$
(6) $q_{i+1} = q_i - r * B * \Delta P_i(T) * \Delta q_i$

Algorithm 1: Set method for parameters ξ and q.

Input: $NIO_i(S_j)$, m_i, k_i, where $i \in (1, l-1)$, $j \in (1, k)$;
(1) **for** $j = 1$ to k **do**
  if $NIO_i(S_j) == 0$**then**
  num++;
(2) $rateArchive = k_i/num$;
(3) $rateAnt = m_i/num$;
(4) $k_{i+1} = C * rateArchive * ENUM$;
(5) $m_{i+1} = C * rateAnt * ENUM$;

Algorithm 2: Set method for parameters m and k.

number of ants needed to produce one non-inferior solution, respectively. Finally, set the new parameters' value to the product of the old parameters' value, the ratio factors, adjustment coefficient C, and expected number $ENUM$.

3. Experiment Results and Discussion

The application field and performance of the algorithm are usually studied by comparing the performance of different MOO algorithms when solving benchmark problems. Referring to some existing mixed-variable MOO algorithms [29–33], this paper designs some problems for algorithm experiments, besides comparing with other well-known MOO algorithms to verify the performance of the algorithm.

3.1. Experimental Environment. The operating environment of the experiment is as follows: Thinkpad T470p computer; Core i7-7700HQ CPU (4cores) * 2; 24 GB memory; 512 GB solid hard disk; and equipped with Windows 10 operating system. The programming tool is Microsoft Visual Studio 2017, and the programming language is C#.

3.2. Benchmark Problem. In this paper, we select eight well-known benchmark problems to evaluate MOO algorithms, that is, Schaffer, Fonseca, Kursawe, ZDT problems, Viennet2, and Viennet3 [34]. These benchmark problems have two (Schaffer, Fonseca, Kursawe, and ZDT family) or three objectives (Viennet2 and Viennet3), and they occupy different properties: separability, unimodality multimodality, convexity, linearity, non-convexity, continuity, discontinuity, bias, Pareto many-to-one, and so on.

The problem name, variable count(N), variable bounds, designed variables, and objective functions are shown in Table 1.

The variables of the eight benchmark problems are all continuous variables. In order to test MOO algorithms with mixed variables, we modify the problems to make some variables as PDV and some variables as RDV; then the continuous problems become mixed problems. PDV and RDV are calculated by the following equations:

$$x_{PDV} \in \left\{x | x_{\mathrm{MIN}} + I * \frac{x_{\mathrm{MAX}} - x_{\mathrm{MIN}}}{N}, \quad I = 0, \ldots, N\right\}, \tag{8}$$

$$x_{RDV} \in \left\{x | x_{\mathrm{MIN}} + I * \frac{x_{\mathrm{MAX}} - x_{\mathrm{MIN}}}{N}, I = RND(N)\right\}, \tag{9}$$

where N is the number of equal divisions of the value range. In order to make the variable a value of 0, N takes a positive even number. $RND(N)$ is a random nonnegative integer not greater than N. In order to make the distribution range of x_{RDV} larger, we need to take every number in $\{0, \ldots, N\}$ once. The domain of x_{PDV} is a set of $N+1$ ordered discrete variables increasing from x_{MIN} to x_{MAX}, and the domain of x_{RDV} is a set of $N+1$ disordered discrete variables between x_{MIN} and x_{MAX}.

If N is large enough, the Pareto set of the mixed problems is similar to the Pareto set of the continuous problems.

3.3. Performance Metrics. Convergence and diversity are usually the two most important criteria for the evaluation of MOO algorithms. The convergence refers to the distance from the non-dominated front generated by the

Table 1: Test benchmark problems.

Problem name	Variable count (N)	Variable bounds	Designed variables	Objective functions
Schaffer	1	$[-10^3, 10^3]$	x_1 is RDV	$f_1 = x^2,\ f_2 = (x-2)^2$
Fonseca	3	$[-4, 4]$	x_2 is PDV x_3 is RDV	$f_1 = 1 - \exp(-\sum_{i=1}^{3}(x_i - (1/\sqrt{3}))^2)$ $f_2 = 1 - \exp(-\sum_{i=1}^{3}(x_i + (1/\sqrt{3}))^2)$
Kursawe	3	$[-5,5]$	x_2 is PDV x_3 is RDV	$f_1 = \sum_{i=1}^{n-1}(-10\exp(-0.2 * \sqrt{x_i^2 + x_{i+1}^2}))\ f_2 = \sum_{i=1}^{n}(\lvert x_i\rvert^{0.8} + 5(\sin\ x_i)^3)$
ZDT1	4	$[0,1]$	x_3 is PDV x_4 is RDV	$f_1 = x_1,\ f_2 = g(x)(1 - \sqrt{f_1/g(x)}),\ g(x) = 1 + 9/(N-1)\sum_{i=2}^{N} x_i$
ZDT2	4	$[0,1]$	x_3 is PDV x_4 is RDV	$f_1 = x_1,\ f_2 = g(x)(1 - (f_1/g(x))^2),\ g(x) = 1 + 9/(N-1)\sum_{i=2}^{N} x_i$
ZDT3	4	$[0,1]$	x_3 is PDV x_4 is RDV	$f_1 = x_1,\ f_2 = g(x)(1 - \sqrt{f_1/g(x)} - f_1/g(x)\sin(10\pi f_1)),$ $g(x) = 1 + 9/(N-1)\sum_{i=2}^{N} x_i$
Viennet2	2	$[-4,4]$	x_2 is PDV	$f_1 = 1/2(x_1 - 2)^2 + 1/13(x_2 + 1)^2 + 3$ $f_2 = 1/36(x_1 + x_2 - 3)^2 + 1/8(x_2 - x_1 + 2)^2 - 17$ $f_3 = 1/175(x_1 + 2x_2 - 1)^2 + 1/17(2x_2 - x_1)^2 - 13$
Viennet3	2	$[-3,3]$	x_2 is PDV	$f_1 = 1/2(x_1^2 + x_2^2) + \sin(x_1^2 + x_2^2),$ $f_2 = 1/8(3x_1 - 2x_2 + 4)^2 + 1/27(x_2 - x_2 + 1)^2 + 15$ $f_3 = 1/(x_1^2 + x_2^2 + 1) - 1.1e^{-(x_1^2 + x_2^2)}$

optimization algorithm to the true Pareto front; the diversity involves coverage area and uniformity; and a front with wide coverage and good uniformity is always pursued.

We have used generational distance (GD) [35] and inverted generational distance plus (IGD^+) [36] for measuring convergence and spread for measuring coverage.

GD: let $T^* = \{F_1^*, \ldots F_i^* \ldots F_{|T^*|}^*\}$ be a set of uniformly distributed Pareto optimal points in the true PF(TPF), and $T = \{F_1, \ldots F_i \ldots F_{|T|}\}$ be a non-dominated front of the problems. The GD of T is the average distance from each solution in T to the nearest reference point:

$$GD(T^*, T) = \frac{1}{|T|}\sum_{j=1}^{|T|}\min_{F_i^* \in T^*} d_{GD}(F_i^*, F_j), \tag{10}$$

where F_i is the objective function corresponding to the solution S_i and $F = (f_1(S_i) \ldots f_i(S_i) \ldots f_v(S_i))$. $d_{GD}(F_i^*, F_j)$ is the Euclidean distance between F_j and F_i^*.

IGD^+: the IGD^+ of T is the average distance from each reference point in T^* to the nearest solution.

$$IGD^+(T^*, T) = \frac{1}{|T^*|}\sum_{i=1}^{|T^*|}\min_{F_j \in T} d_{IGD^+}(F_i^*, F_j). \tag{11}$$

In IGD^+, the distance between a reference point $F^* = (f_1^*, f_2^*, \ldots, f_v^*)$ and a solution $F = (f_1, f_2, \ldots, f_v)$ is calculated in the objective space for the v-objective minimization problem as follows:

$$d_{IGD^+}(F^*, F) = \sqrt{\sum_{i=1}^{v}(\max\{f_i - f_i^*, 0\})^2}. \tag{12}$$

Generalized Spread (see [36]). The generalized spread is an indicator that measures the distribution and spread of the obtained non-dominated front of the problems with two or more objectives:

$$\Delta(T^*, T) = \frac{\sum_{i=1}^{v} d(e_i, T) + \sum_{X \in T^*}|d(X, T) - \overline{d}|}{\sum_{i=1}^{v} d(e_i, T) + |T^*|\overline{d}}, \tag{13}$$

where $\{e_1, e_2, \ldots, e_m\}$ are m extreme solutions in T^* and

$$d(X, T) = \min_{Y \in T, Y \neq X}\|F(X) - F(Y)^2\|,$$
$$\overline{d} = \frac{1}{|T^*|}\sum_{X \in T^*} d(X, T). \tag{14}$$

3.4. Performance Improvement of SAMOACO$_{MV}$. In order to test the performance of SAMOACO$_{\mathrm{MV}}$, some experiments are carried out under the same conditions, for example, when the problem is the modified Fonseca problem, the maximum number of algorithm iterations is the same. Table 2 lists the setting schemes of the algorithm parameters in the six experiments. The first five experiments test the performance of the MOACO$_{\mathrm{MV}}$ algorithm that have different parameter values of ξ and q and the same values of m and k. The sixth experiment tests the performance of the SAMOCO$_{\mathrm{MV}}$ algorithm.

Table 3 shows the performance of the 6 experiments for the Fonseca test problem. For each major cell of Table 3, the first column indicates the mean of 25 runs, the second column indicates the standard deviation, and the third column indicates the rank.

Figure 2 shows the Pareto points obtained with reference to the true Pareto frontier graphically using results from 1 of the 25 runs. MOACO$_{\mathrm{MV5}}$ generates only a few Pareto points, so it is not shown in the figure.

It can be seen from the figure and the table:

(i) The figure shows that the Pareto points generated by the SAMOACO$_{\mathrm{MV}}$ algorithm are right on TPF, and the table shows the overall rank value of the

Table 2: Deployment of algorithms' parameters.

	m	k	ξ	q
$MOACO_{MV}1$	50	200	0.001	0.001
$MOACO_{MV}2$	50	200	0.01	0.01
$MOACO_{MV}3$	50	200	0.1	0.1
$MOACO_{MV}4$	50	200	1	1
$MOACO_{MV}5$	50	200	10	10
$SAMOACO_{MV}$	Self-adaptive control			

Table 3: The performance of six experiments for the Fonseca problem.

Fonseca	Generational distance			Inverted generational distance			Generalized spread			Sum of ranks	Overall rank
	Mean	Stdev	Rank	Mean	Stdev	Rank	Mean	Stdev	Rank		
$MOACO_{MV}1$	0.0167	0.0118	5	0.2153	0.0893	5	1.1378	0.1869	6	16	6
$MOACO_{MV}2$	0.0031	0.0014	4	0.1545	0.0872	4	0.9836	0.0605	5	13	4
$MOACO_{MV}3$	0.0001	1.1788	2	0.0096	0.0103	3	0.6524	0.0561	2	7	3
$MOACO_{MV}4$	0.0002	1.32E-05	3	0.0035	0.0004	2	0.5147	0.0282	1	6	1
$MOACO_{MV}5$	0.0815	0.0323	6	0.246	0.0788	6	0.6905	0.2014	3	15	5
$SAMOACO_{MV}$	9.45E-05	1.24E-05	1	0.0026	0.0003	1	0.7064	0.0384	4	6	1

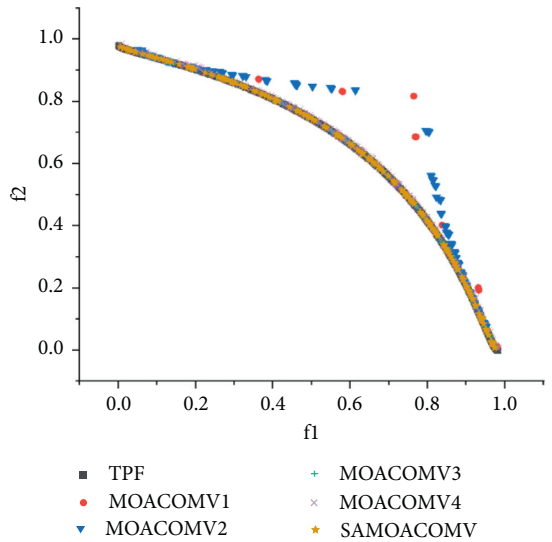

Figure 2: Pareto front for Fonseca problem.

$SAMOACO_{MV}$ algorithm is minimum, that is, the performance of the $SAMOACO_{MV}$ algorithm is the best in all experiments.

(ii) When the $MOACO_{MV}$ algorithm adopts setting schemes 3 and 4, the algorithm performance is basically the same as that of the $SAMOCO_{MV}$ algorithm, but when other schemes are used, the algorithm performance is very poor, which shows that the performance of the $MOACO_{MV}$ algorithm relies heavily on parameter settings.

More experiments show that the performance of the $SAMOCO_{MV}$ algorithm is better than that of the $MOACO_{MV}$ algorithm; especially, this advantage is more obvious when the values of m and k are small.

3.5. Performance Comparison Using Benchmark Problems. In order to test the performance of the algorithm, this paper compares the $SAMOCO_{MV}$ algorithms with the well-known MOO algorithm NSGAII, SPEA2, SMPSO, MOEAD, NSGAIII, and MOEA/D-IEpsilon. These algorithm programs come from jMetal [30], and the two algorithms can only be used to deal with CV MOO.

In order to compare the multiobjective optimization algorithms, each algorithm is allowed to run for the test problems for a constant number of function evaluations. The performance metrics are calculated for each algorithm run. This procedure is repeated for 20 runs, and the mean and standard deviation of the performance metrics are recorded for each algorithm.

3.5.1. Results Based on Schaffer, Fonseca, and Kursawe Problems. Tables 4–6 show the mean and standard deviation of generational distance, inverted generational distance plus, and generalized spread for different algorithms, respectively. The $SAMOACO_{MV}$ fetches good performance metric values in terms of the Schaffer problem, while other algorithms cannot obtain or only obtain a few Pareto points. It may be because the only variable of Schaffer problem is changed to a discrete variable, and other algorithms cannot solve the pure discrete variable problem. For the Fonseca and Kursawe problems, compared with other techniques, $SAMOACO_{MV}$ obtains excellent GD and IGD^{+} values, only slightly weaker than MOEA/D-IEpsilon, but obtains relatively poor generalized spread value.

Figures 3–5 provide a graphical visualization of the Pareto points obtained for Schaffer, Fonseca, and Kursawe problems, respectively. For the Schaffer problem, none of the other algorithms apart from $SAMOACO_{MV}$ was able to produce any Pareto points close to the TPF. For the Fonseca

Table 4: Mean and standard deviation of generational distance.

		Schaffer	Fonseca	Kursawe	ZDT1	ZDT2	ZDT3	Viennet2	Viennet3	Sum of ranks	Overall rank
NSGAII	Mean	—	0.0004	0.0005	0.0017	0.0029	0.0008	0.0008	0.0002	29	4
	Stdev	—	3.51E-05	8.89E-05	0.0003	0.0006	0.0002	0.0003	3.72E-05		
	Rank	2	4	2	5	5	3	4	4		
SPEA2	Mean	—	0.0004	0.0005	0.0021	0.0036	0.0017	0.0008	0.0003	39	6
	Stdev	—	4.82E-05	0.0001	0.001	0.0022	0.0018	0.0001	0.0001		
	Rank	2	4	2	7	7	7	4	6		
SMPSO	Mean	—	0.0004	0.0012	0.0011	0.0019	0.0009	0.001	0.0002	32	5
	Stdev	—	3.92E-05	0.0004	0.0001	0.0003	0.0002	0.0003	9.86E-05		
	Rank	2	4	6	3	3	4	6	4		
MOEAD	Mean	—	0.0003	0.0015	0.0014	0.0024	0.0011	—	—	39	6
	Stdev	—	4.00E-05	0.0007	0.0002	0.0003	0.0002	—	—		
	Rank	2	3	7	4	4	5	7	7		
NSGAIII	Mean	—	0.0004	0.0005	0.0019	0.0032	0.0012	1.77E-05	1.15E-05	28	3
	Stdev	—	3.82E-05	0.0001	0.0007	0.0009	0.0008	4.61E-06	4.77E-07		
	Rank	2	4	2	6	6	6	1	1		
MOEA/D-IEpsilon	Mean	—	9.93E-05	0.0002	0.0005	0.0004	0.0001	0.0003	4.37E-05	17	2
	Stdev	—	2.76E-06	1.79E-05	8.28E-05	3.33E-05	1.23E-05	7.56E-05	7.48E-06		
	Rank	2	2	1	2	2	2	3	3		
$SAMOACO_{MV}$	Mean	5.38E-05	9.68E-05	0.0005	0.0001	0.0002	8.25E-05	3.21E-05	2.12E-05	11	1
	Stdev	9.01E-05	3.23E-06	7.08E-05	6.29E-06	6.63E-06	4.86E-06	1.54E-06	3.03E-06		
	Rank	1	1	2	1	1	1	2	2		

Table 5: Mean and standard deviation of inverted generational distance plus.

		Schaffer	Fonseca	Kursawe	ZDT1	ZDT2	ZDT3	Viennet2	Viennet3	Sum of ranks	Overall rank
NSGAII	Mean	—	0.0048	0.0044	0.0156	0.0418	0.0078	0.0141	0.0052	37	5
	Stdev	—	0.0005	0.0005	0.0055	0.0674	0.003	0.0026	0.0006		
	Rank	2	6	3	7	5	4	6	4		
SPEA2	Mean	—	0.0049	0.0048	0.0150	0.0696	0.0086	0.0066	0.0041	39	6
	Stdev	—	0.0004	0.0005	0.0017	0.1054	0.0011	0.0005	0.0004		
	Rank	2	7	4	6	7	6	4	3		
SMPSO	Mean	—	0.0046	0.0123	0.0098	0.0128	0.0074	0.0135	0.0053	32	4
	Stdev	—	0.0002	0.0021	0.0008	0.0013	0.0010	0.0024	0.0006		
	Rank	2	4	7	3	3	3	5	5		
MOEAD	Mean	—	0.0042	0.008	0.0131	0.0188	0.0101	—	—	39	6
	Stdev	—	0.0004	0.0009	0.0016	0.0022	0.0019	—	—		
	Rank	2	3	5	4	4	7	7	7		
NSGAIII	Mean	—	0.0046	0.0089	0.0149	0.0622	0.0079	0.0005	0.0005	30	3
	Stdev	—	0.0004	0.003	0.0049	0.0944	0.0040	9.83E-06	3.64E-05		
	Rank	2	4	6	5	6	5	1	1		
MOEA/D-IEpsilon	Mean	—	0.0015	0.0031	0.0097	0.0058	0.0030	0.0062	0.0064	19	2
	Stdev	—	5.32E-05	0.0003	0.0017	0.0004	0.0002	0.0006	0.0007		
	Rank	2	1	1	2	2	2	3	6		
$SAMOACO_{MV}$	Mean	5.32E-06	0.0017	0.0040	0.0021	0.0029	0.0016	0.0007	0.0005	11	1
	Stdev	2.38E-06	7.78E-05	0.0003	6.90E-05	0.0001	0.0001	7.19E-06	4.73E-05		
	Rank	1	2	2	1	1	1	2	1		

Table 6: Mean and standard deviation of generalized spread.

		Schaffer	Fonseca	Kursawe	ZDT1	ZDT2	ZDT3	Viennet2	Viennet3	Sum of ranks	Overall rank
NSGAII	Mean	—	0.2297	0.3980	0.5673	0.6321	0.5606	0.4579	0.4094	22	1
	Stdev	—	0.0311	0.0383	0.0649	0.1363	0.0703	0.0356	0.0335		
	Rank	2	3	1	4	5	2	3	2		
SPEA2	Mean	—	0.1948	0.4531	0.6467	0.7259	0.7707	0.2058	0.5898	31	4
	Stdev	—	0.0184	0.0529	0.0637	0.1715	0.1301	0.0247	0.0207		
	Rank	2	1	2	6	7	7	1	5		
SMPSO	Mean	—	0.2306	0.8257	0.4907	0.5718	0.5416	0.3143	0.2296	22	1
	Stdev	—	0.0156	0.0610	0.0437	0.0966	0.0528	0.0276	0.0269		
	Rank	2	4	6	2	4	1	2	1		
MOEAD	Mean	—	0.2958	0.5525	0.4562	0.4961	0.6611	—	—	33	6
	Stdev	—	0.0416	0.0561	0.0465	0.0540	0.0484	—	—		
	Rank	2	5	4	1	1	6	7	7		
NSGAIII	Mean	—	0.3346	0.8579	0.5899	0.7028	0.626	0.6129	0.9181	42	7
	Stdev	—	0.0355	0.0783	0.0729	0.1438	0.1078	0.0146	1.23E-02		
	Rank	2	6	7	5	6	5	5	6		
MOEA/D-IEpsilon	Mean	—	0.2166	0.5719	0.8105	0.5029	0.5857	0.8131	0.5781	32	5
	Stdev	—	0.0107	0.0527	0.0957	0.0440	0.0470	0.0583	0.0934		
	Rank	2	2	5	7	2	4	6	4		
$SAMOACO_{MV}$	Mean	0.0958	0.4827	0.4747	0.5028	0.5278	0.5607	0.5403	0.4938	27	3
	Stdev	0.0006	0.0254	0.0447	0.0203	0.0166	0.0207	0.0099	0.0093		
	Rank	1	7	3	3	3	3	4	3		

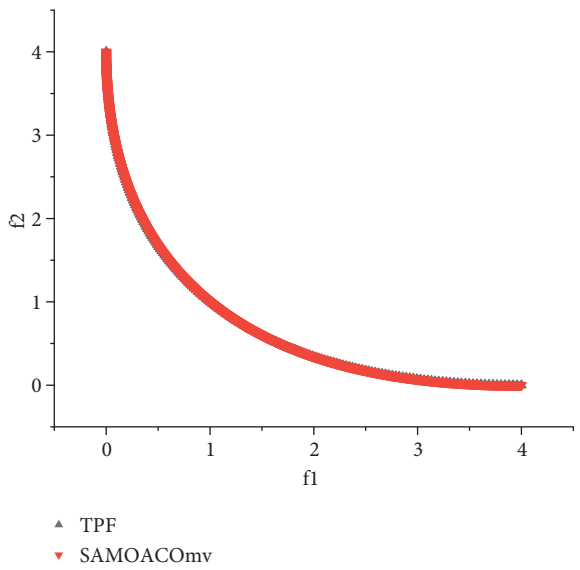

Figure 3: Pareto front for the Schaffer problem.

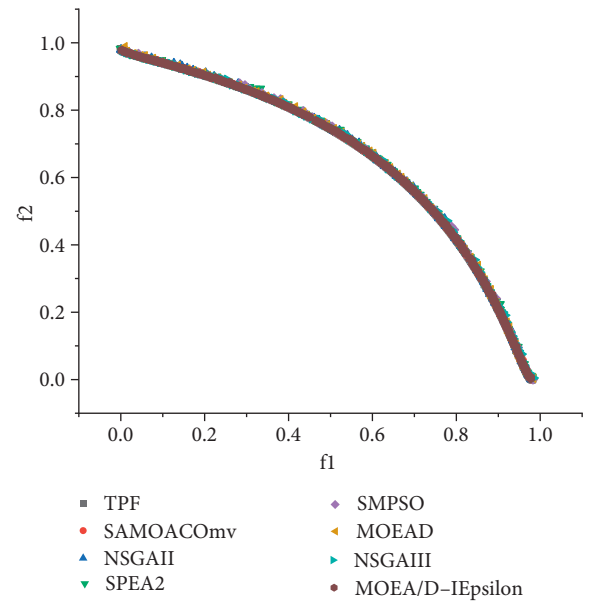

Figure 4: Pareto front for the Fonseca problem.

problem, the performance of each algorithm is very good, and the generated Pareto points right on TPF. For the Kursawe problem, the performance of each algorithm is also very good, except that some Pareto points generated by SMPSO and MOEAD deviate slightly from TPF.

3.5.2. Results Based on ZDT (ZDT1–ZDT3) Problems. From Tables 4 and 5, $SAMOACO_{MV}$ ranks 1 for *ZDT* problems, which means that the $SAMOACO_{MV}$ outperforms other algorithms on the performance metrics *GD* and IGD^{+}. From Table 6, $SAMOACO_{MV}$ performed slightly worse on generalized spread for *ZDT* problems, ranking 3.

From Figures 6–8, all the algorithms have good performance, and the obtained Pareto front is basically consistent with the TPF. Some algorithms do not perform well on certain problems, such as SPEA2 and MOEAD produce some points that deviate slightly from the TPF for *ZDT1* and *ZDT2* problems.

3.5.3. Results Based on Viennet2 and Viennet3 Problems. As shown in Table 4, the mean of *GD* of $SAMOACO_{MV}$ for Viennet2 and Viennet3 problems are about 0.000032 and

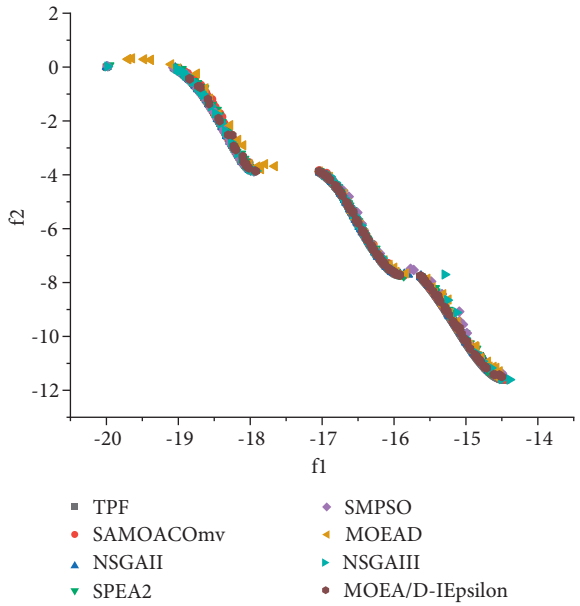

Figure 5: Pareto front for the Kursawe problem.

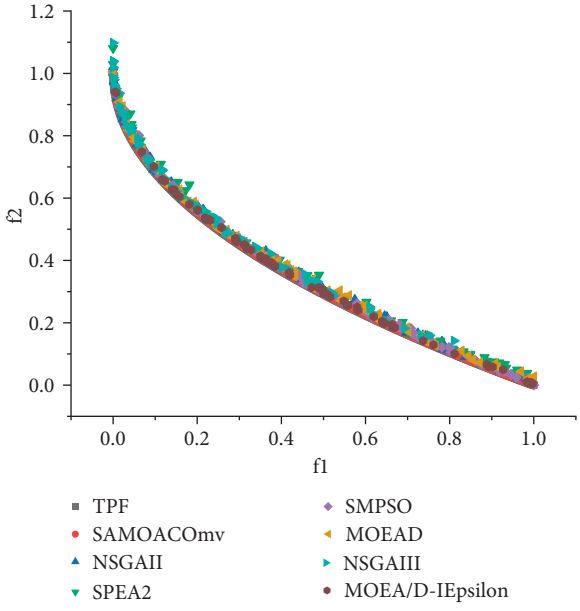

Figure 6: Pareto front for the ZDT1 problem.

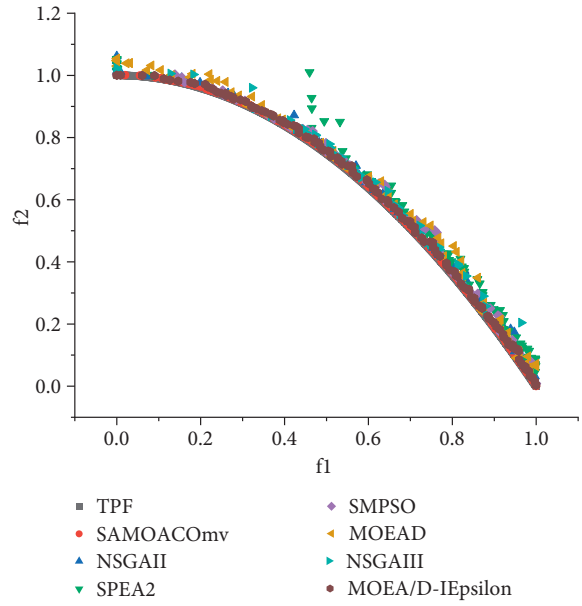

Figure 7: Pareto front for the ZDT2 problem.

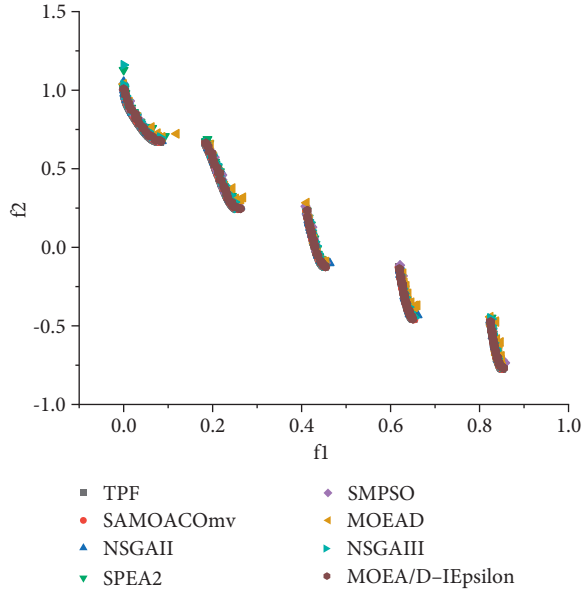

Figure 8: Pareto front for the ZDT3 problem.

0.000021, respectively, which are only slightly worse than the mean values of NSGAIII but far better than the corresponding performance metric values of other algorithms. It can be seen from Table 5 that similar to *GD*, the SAMOACO$_{MV}$ has almost the best IGD^+ mean for Viennet2 and Viennet3 problems, around 0.0007 and 0.0005, respectively, only slightly worse than the mean of NSGAIII. From Table 6, SAMOACO$_{MV}$ performed worse on generalized spread for Viennet2 and Viennet3 problems, ranking 4 and 3, respectively.

In Figures 9 and 10, the approximated Viennet2 and Viennet3 fronts of each algorithm are shown. It is clear that SAMOACO$_{MV}$ obtained much more Pareto points, they converge well to the TPF, and they widely and uniformly distribute along the TPF, which illustrates that it has better convergence and diversity compared with the other algorithms.

In summary, with *GD*, IGD^+, and generalized spread taken into consideration, SAMOACO$_{MV}$ is quite a competitive algorithm in terms of the convergence of the generated Pareto solution set; the overall rank is 1. But SAMOACO$_{MV}$ is slightly weaker than other algorithms in the coverage performance; the overall rank is 3.

4. Experiment Results on Spring Design Problem

The spring design problem is a common engineering practice problem and widely used MOO algorithm

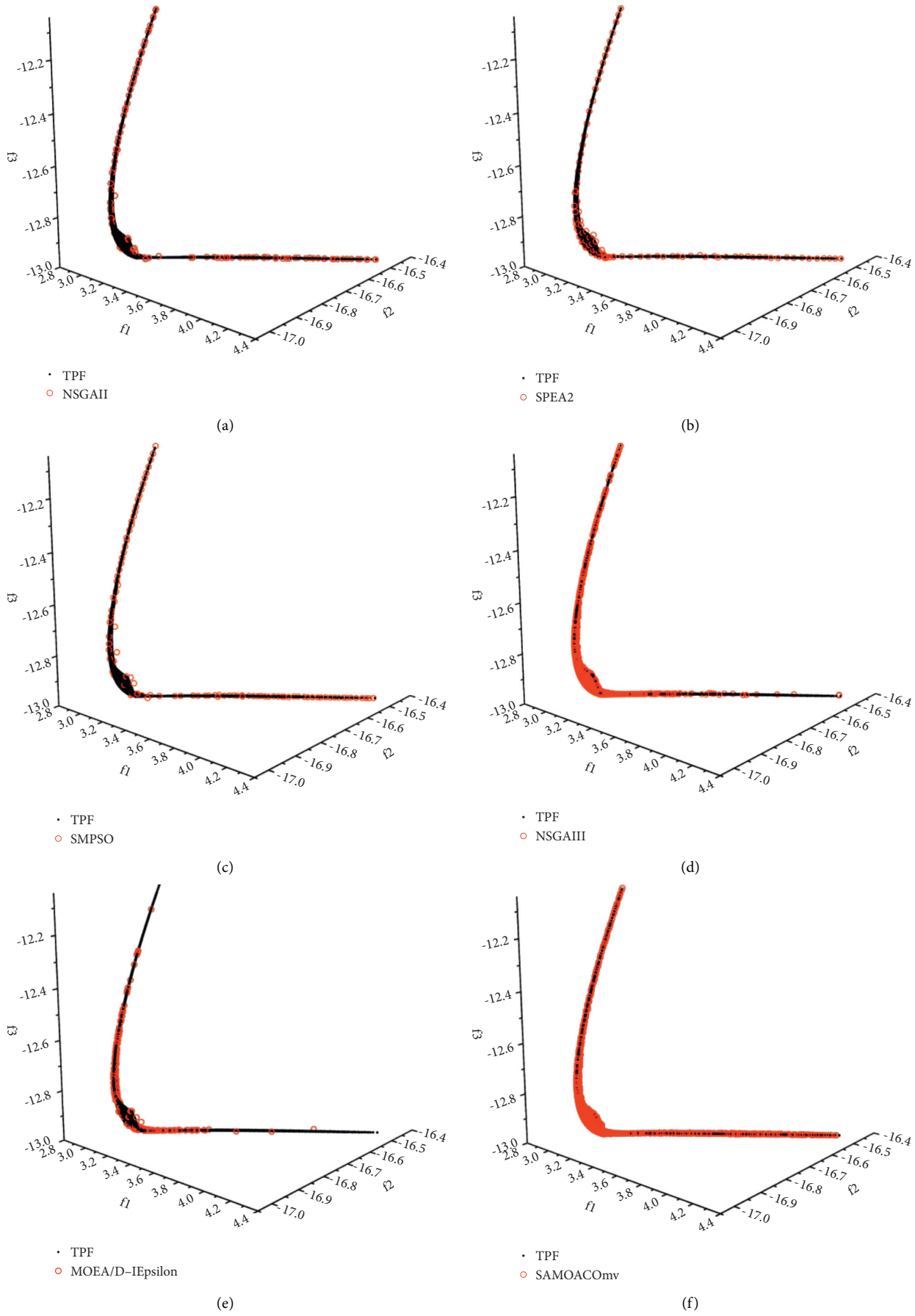

Figure 9: Pareto front for the Viennet2 problem.

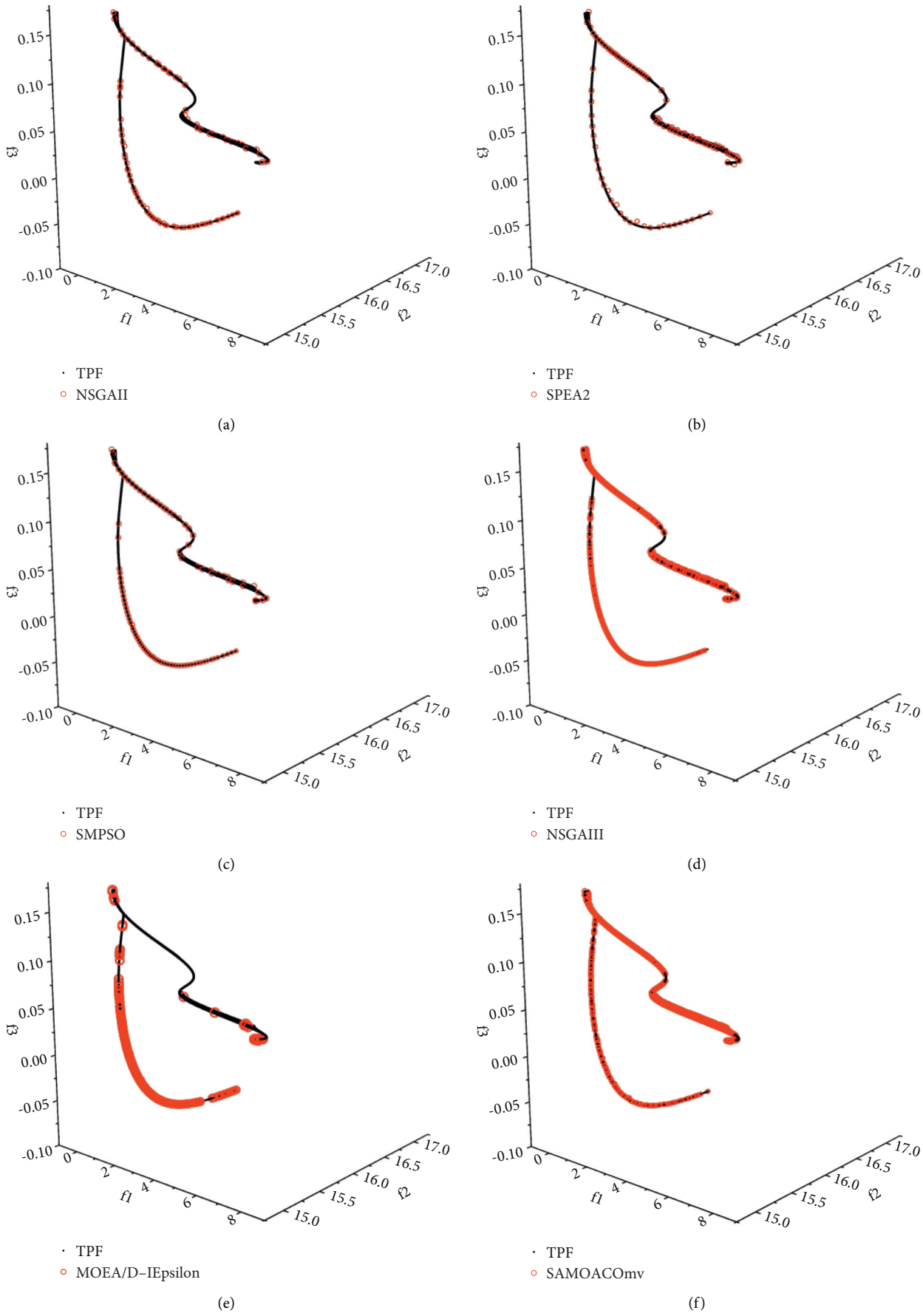

Figure 10: Pareto front for the Viennet3 problem.

performance verification example [37, 38], and it is a mixed-variable MOO problem containing continuous and discrete variables. We use the spring design problem to test the performance of the SAMOACO$_{MV}$ algorithm in this paper.

4.1. Problem Description. The spring design problem consists of two discrete variables and one continuous variable. The objectives are to minimize the volume of the spring and minimize the stress developed by applying a load. Variables are the diameter of the wire (d), the diameter of the spring (D), and the number of turns (N). Denoting the variable vector $\vec{x} = (x_1, x_2, x_3) = (N, d, D)$, formulation of this problem with two objectives and eight constraints is as follows [38]:

$$\begin{aligned}
&\text{minimize } f_1(\vec{x}) = 0.25\pi^2 x_2^2 x_3 (x_1 + 2)\\
&\text{maximize } f_2(\vec{x}) = \frac{8KP_{\max}x_3}{\pi x_3^2}\\
&g_1(\vec{x}) = l_{\max} - \frac{P_{\max}}{k} - 1.05(x_1 + 2)x_2 \geq 0\\
&g_2(\vec{x}) = x_2 - d_{\min} \geq 0,\\
&g_3(\vec{x}) = D_{\max} - (x_2 + x_3) \geq 0,\\
&g_4(\vec{x}) = C - 3 \geq 0,\\
&\text{Subject to}\\
&g_5(\vec{x}) = \delta_{pm} - \delta_p \geq 0,\\
&g_6(\vec{x}) = \frac{P_{\max} - P}{k} - \delta_w \geq 0,\\
&g_7(\vec{x}) = S - \frac{8KP_{\max}x_3}{\pi x_2^3} \geq 0,\\
&g_8(\vec{x}) = V_{\max} - 0.25\pi^2 x_2^2 x_3 (x_1 + 2) \geq 0,
\end{aligned} \tag{15}$$

where x_1 is an integer, x_2 is a discrete variable, and x_3 is a continuous variable.

The parameters used are as follows:

$$\begin{aligned}
K &= \frac{4C - 1}{4C - 4} + \frac{0.615x_2}{x_3},\\
P &= 300\text{lb},\\
D_{\max} &= 3\text{ in}, k = \frac{Gx_2^4}{8x_1x_3^3},\\
P_{\max} &= 1000\text{lb},\\
\delta_w &= 1.25\text{in}, \delta_p = \frac{P}{k},\\
l_{\max} &= 14\text{in},\\
\delta_{pm} &= 6\text{in},\\
S &= 189\text{ ksi},\\
d_{\min} &= 0.2\text{in},\\
C &= \frac{x_3}{x_2},\\
G &= 11,500,000\frac{\text{lb}}{\text{in}^2}, V_{\max} = 30\text{in}^3.
\end{aligned} \tag{16}$$

The 42 discrete values of d are given below:

$$d = \begin{pmatrix}
0.009, & 0.0095, & 0.0104, & 0.0118, & 0.0128, & 0.0132,\\
0.014, & 0.015, & 0.0162, & 0.0173, & 0.018, & 0.020,\\
0.023, & 0.025, & 0.028, & 0.032, & 0.035, & 0.041,\\
0.047, & 0.054, & 0.063, & 0.072, & 0.080, & 0.092,\\
0.105, & 0.120, & 0.135, & 0.148, & 0.162, & 0.177,\\
0.192, & 0.207, & 0.225, & 0244, & 0.263, & 0.283,\\
0.307, & 0.331, & 0.362, & 0.394, & 0.4375, & 0.5.
\end{pmatrix}. \tag{17}$$

5. Experiment Results

From Table 7, the mean of GD, IGD^+, and generalized spread of SAMOACO$_{MV}$ for the spring design problem are about 0.0014, 0.064, and 0.3532, respectively, much smaller than other algorithms. The values of the three performance metrics of SAMOACO$_{MV}$ for the spring design problem are all ranked first, and its overall rank is also the first, which shows that SAMOACO$_{MV}$ is optimal in convergence and coverage.

Table 7: The performance for spring design problem.

Spring design	Generational distance			Inverted generational distance			Generalized spread			Sum of ranks	Overall rank
	Mean	Stdev	Rank	Mean	Stdev	Rank	Mean	Stdev	Rank		
NSGAII	0.0119	0.0157	3	0.0225	0.0802	4	0.9629	0.1547	4	11	4
SPEA2	0.0043	0.0027	2	0.0173	0.0399	3	0.9742	0.1291	5	10	2
SMPSO	0.0161	0.0154	4	0.2132	0.1329	5	0.8366	0.1505	2	11	4
GDE3	0.0885	0.1489	5	0.0068	0.0053	2	0.8876	0.1654	3	10	2
$SAMOACO_{MV}$	0.0014	0.0007	1	0.0064	0.0027	1	0.3532	0.0338	1	3	1

Table 8: Pareto points for spring design problem.

Spring design	Number of archive points			Number of Pareto points			Percentage of Pareto points in archive			Sum of ranks	Overall rank
	Mean	Stdev	Rank	Mean	Stdev	Rank	Mean	Stdev	Rank		
NSGAII	43.95	30.1531	3	4.5	5.3852	3	0.0775	0.0656	3	9	3
SPEA2	100	0	1	8.15	7.6796	1	0.0815	0.0768	2	4	1
SMPSO	9.7	0.0897	5	0.15	0.4894	5	0.0131	0.0403	5	15	5
GDE3	16.3	13.0307	4	0.45	0.887	4	0.0254	0.0513	4	12	4
$SAMOACO_{MV}$	55.65	6.483	2	6.35	7.4288	2	0.1044	0.1099	1	5	2

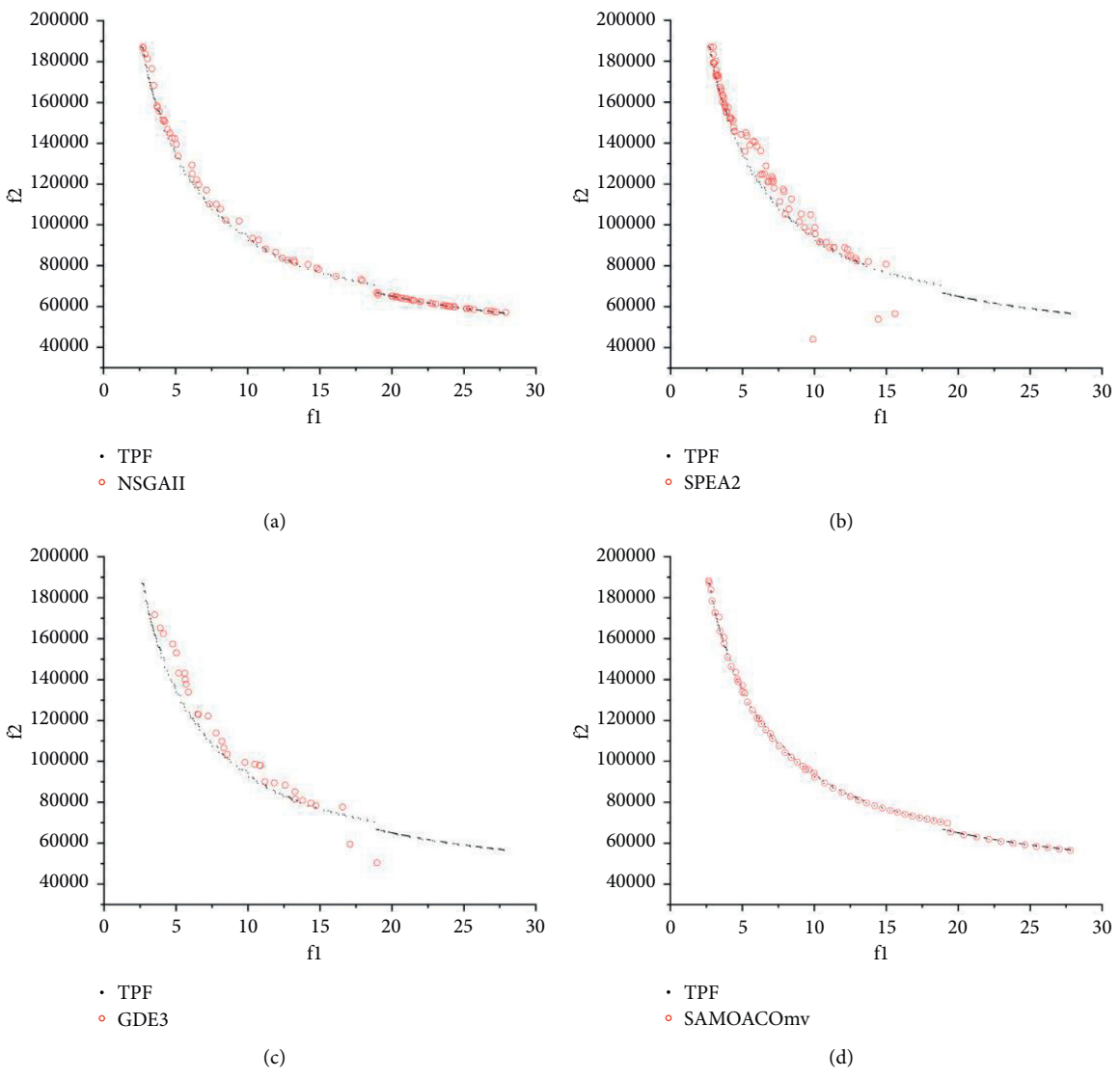

Figure 11: Pareto front for the spring design problem.

As shown in Table 8, for the spring design problem, the number of archive points and the number of Pareto points of $SAMOACO_{MV}$ rank 2, and the percentage of Pareto points in archive ranks 1, which means that $SAMOACO_{MV}$ has the highest comprehensive efficiency in finding Pareto points.

The obtained Pareto frontier is plotted in Figure 11. The TPF represents the set of non-inferior solutions obtained by merging all experimental results from all independent runs of all algorithms and removing the inferior solution. SMPSO can only obtain a few Pareto points, so it is not shown by the figure. It can be seen from Figure 11 that many points of NSGA-II, SPEA2, and GDE3 do not converge to the TPF, and some points of SPEA2 and GDE3 are far away from TPF. The Pareto points of NSGA-II, SPEA2, and GDE3 have poor distributions, and SPEA2 and GDE3 only cover part of TPF. In contrast, the Pareto points obtained by $SAMOACO_{MV}$ widely and uniformly distributed along the TPF, which illustrates that it has better convergence and diversity compared with the other algorithms.

6. Conclusion

In this work, we have modified the single-objective optimization algorithm ACO_{MV} to handle mixed-variable MOO problems and proposed a self-adaptive parameter-setting scheme. Then the performance of $SAMOACO_{MV}$ was thoroughly tested using a set of performance metrics with a well-designed benchmark test suite. Its performance was compared with the state-of-the-art multiobjective optimization algorithms. For all benchmark problems, the $SAMOACO_{MV}$ algorithm has good convergence performance, and its GD and IGD^+ are almost the best. However, the generalized spread of $SAMOACO_{MV}$ is slightly worse, which means that the coverage performance of $SAMOACO_{MV}$ is slightly weaker than other algorithms. For spring design problem, the $SAMOACO_{MV}$ algorithm can get widely and uniformly distributed Pareto front, and it has the best convergence and coverage performance.

In general, the $SAMOACO_{MV}$ algorithm is an excellent MOO algorithm, which adds a new choice for solving MOO problems.

Data Availability

Some or all data, models, or codes that support the findings of this study are available from the corresponding author upon reasonable request.

Disclosure

The approach proposed in this paper has been published at the 2020 IEEE International Congress on Cybermatics (iThings/GreenCom/CPSCom/SmartData/Blockchain-2020) [39]. Based on the conference paper, this paper mainly expands as follows: a new congestion degree of the solution is defined to rank the solutions in the archive, modified the self-adaptive strategy to set the parameters m and k of the $SAMOACO_{MV}$ algorithm, and designed some new mixed-variable MOO benchmark problems to test and compare the performance of the $SAMOACO_{MV}$ algorithm. New performance metrics such as *GD*, IGD^+, and *Generalized Spread* are used to evaluate the performance of the algorithms. All experiments are redone, and the corresponding described text, figure, and table of experimental results are updated.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by National Key Research and Development Program of China (Grant no. 2018YFC1405700) and Industry University Research Cooperation Project of Jiangsu Province (Grant no. BY2019005).

References

[1] Q. Liu, R. Mo, X. Xu, and M Xu, "Multi-objective resource allocation in mobile edge computing using PAES for Internet of Things," *Wireless Networks*, pp. 1–13, 2020.

[2] M. Huang, Q. Zhai, Y. Chen, S. Feng, and F. Shu, "Multi-objective whale optimization algorithm for computation offloading optimization in mobile edge computing," *Sensors*, vol. 21, no. 8, p. 2628, 2021.

[3] G. Fan, L. Chen, H. Yu, and W. Qi, "Multi-objective optimization of container-based microservice scheduling in edge computing," *Computer Science and Information Systems*, vol. 18, 2020.

[4] X. Xu, R. Gu, F. Dai, L. Qi, and S. Wan, "Multi-objective computation offloading for internet of vehicles in cloud-edge computing," *Wireless Networks*, vol. 26, no. 11, 2020.

[5] C. García-Martínez, O. Cordón, and F. Herrera, "A taxonomy and an empirical analysis of multiple objective objective ant colony optimization algorithms for the Bi-criteria TSP," *European Journal of Operational Research*, vol. 180, no. 1, pp. 116–148, 2004.

[6] C. A. Coello, D. A. V. Veldhuizen, and G. B. Lamant, *Evolutionary Algorithms for Solving Multiobjective Problems*, Kluwer Academic Publishers, NY, USA, 2002.

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

[8] E. Zitzler and M. Laumanns, "SPEA2: improving the strength Pareto evolutionary algorithm," TIK-Report, vol. 103, 2001, https://doi.org/10.3929/ethz-a-004284029.

[9] J. Knowles and D. Corne, "The Pareto archived evolution strategy: a new baseline algorithm for multiobjective optimization," *NJ*, IEEE Press, in *Proceedings of the 1999 Congress on Evolutionary Computation,Piscataway*, pp. 9–105, IEEE Press, Washington, July 1999.

[10] 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, 2014.

[11] A. J. Nebro, J. J. Durillo, J. Garcia-Nieto, C. Coello, and E. Alba, "SMPSO: a new PSO-based metaheuristic for multi-objective optimization," in *Proceedings of the Computational Intelligence in Miulti-Criteria Decision-Making*, March 2009.

[12] M. R. Sierra and C. C. A. Coello, "Improving PSO-based multi-objective optimization using crowding, mutation and e-dominance," in *International Conference on Evolutionary Multi-Criterion Optimization Springer*, Berlin, Germany, 2005.

[13] "Gde3, "The Third Evolution Step of Generalized Differential Evolution," in *Proceedings of the 2005 Congress on Evolutionary Computation (CEC 2005)*, pp. 443–450, Sept 2005, http://www.iitk.ac.in/kangal/papers/k2005013.pdf.

[14] H. Li and Q. F. Zhang, "Multi-objective optimization problems with complicated Pareto sets," *MOEA/D and NSGA-II*, 2008.

[15] Z. Fan, W. Li, X. Cai et al., "An improved epsilon constraint-handling method in MOEA/D for CMOPs with large infeasible regions," *Soft Computing*, vol. 23, 2017.

[16] C. E. Mariano, E. Morales, and P Cuauhnahuac, "A multiple objective ant-Q algorithm for the design of water distribution irrigation networks," *Technical Report HC-9904, Instituto Mexicano de Tecnología del Agua, Progreso, Mexico*, 1999, http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.46.7622.

[17] K. Doerner, W. J. Gutjahr, R. F. Hartl, C Strauss, and C Stummer, "Pareto Ant Colony Optimization: A metaheuristic approach to multiobjective portfolio selection," *Annals of Operations Research*, vol. 131, 2004.

[18] B. Barán and M. Schaerer, "A multiobjective ant colony system for vehicle routing problem with time Windows. Proc," in *Proceedings of the 21st IASTED International Conference on Applied Informatics*, no. 2, pp. 97–102, Insbruck, Austria, January 2003.

[19] P. Cardoso and M. Jesús, "A.Márquez," *MONACO-Multi-Objective Network Optimisation Based on an ACO. Proc.X Encuentros de Geometría Computacional*, Spain, Seville, 2003.

[20] V. T'kindt, N. Monmarché, F. Tercinet, and D. Laügt, "An Ant Colony Optimization algorithm to solve a 2-machine bicriteria flowshop scheduling problem," *European Journal of Operational Research*, vol. 142, no. 2, pp. 250–257, 2002.

[21] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[22] J. I. N. Bingyao, "An introduction of some new optimum techniques," *Bulletin of Science and Technology*, vol. 16, no. 3, pp. 119–124, 2000.

[23] J. A. Manson, T. W. Chamberlain, and R. A. Bourne, "MVMOO: mixed variable multi-objective optimisation," *Journal of Global Optimization*, vol. 80, no. 4, pp. 865–886, 2021.

[24] H. Li, Z. Liu, and P. Zhu, "An improved multi-objective optimization algorithm with mixed variables for automobile engine hood lightweight design," *Journal of Mechanical Science and Technology*, vol. 35, no. 5, pp. 2073–2082, 2021.

[25] Z. O. Khokhar, H. Vahabzadeh, A. Ziai, G. G. Wang, and C. Menon, "On the performance of the PSP method for mixed-variable multi-objective design optimization." ASME. J. Mech. Des," vol. 132, no. 7, Article ID 071009, 2010.

[26] S. Liao, K. Socha, M. A. Montes de Oca, T. Stützle, and M. Dorigo, "Ant colony optimization for mixed-variable optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 503–518, 2014.

[27] V. Chankong and T Haimes, "Multio- bjective decision making: theory and methodology," North Holland, New York, 1983.

[28] Y. Fei, N. Li, and Z. Han, "Multi-objective optimization method and its application based on Pareto sets," *Hoisting and Conveying Machinery*, no. 9, pp. 13–15, 2006.

[29] F. Kursawe, "A Variant of Evolution Strategies for Vector Optimization," in *Parallel Problem Solving from Nature*, pp. 193–197, Springer, Berlin, 1991.

[30] J. Antonio and J. D Nebro, "jMetal 5.10 [CP]," 2021, https://github.com/jMetal/jMetal.

[31] M. K. Rahman, "An intelligent moving object optimization algorithm for design problems with mixed variables, mixed constraints and multiple objectives," *Structural and Multidisciplinary Optimization*, vol. 32, no. 1, pp. 40–58, 2006.

[32] Y. Xiong, "Mixed discrete fuzzy nonlinear programming for engineering design optimization," *Department of Mechanical Engineering*, pp. 109–111, University of Miami, Coral Gables, Florida, 2002.

[33] Z. Xuejun and P. Deng, "Multiobjective Optimization Design with Mixed- Discrete Variables in Mechanical Engineering via Pareto Genetic Algorithm," *Journal of Shanghai Jiaotong University*, vol. 34, no. 3, pp. 411–414, 2000.

[34] L. Yin, M. Zhuang, J. Jia, and H. Wang, "Energy saving in flow-shop scheduling management: an improved multi-objective model based on grey wolf optimization algorithm," *Mathematical Problems in Engineering*, vol. 2020, Article ID 9462048, 14 pages, 2020.

[35] D. A. Van, V. Gary, and B. Lamont, "Multiobjective evolutionary algorithm research: a history and analysis," *Evolutionary Computation*, vol. 8, no. 2, 1998.

[36] H. Ishibuchi, H. Masuda, and Y. Nojima, "A study on performance evaluation ability of a modified inverted generational distance indicator," pp. 695–702, 2015.

[37] Z. O. Khokhar, H. Vahabzadeh, A. Ziai, and W Gary, "On the performance of the PSP method for mixed-variable multi-objective design optimization," *Journal of Mechanical Design*, vol. 132, no. 7, Article ID 071009, 2010.

[38] K. Deb and A. Srinivasan, "Innovization: innovating design principles through optimization," in *Proceedings of the Genetic and evolutionary computation conference, GECCO*, Proceedings, Seattle, Washington, USA, July 2006.

[39] Y. Gong, W. Wang, and S. Gong, "Research of a self-adaptive mixed-variable multi-objective ant colony optimization algorithm," in *Proceedings of the 2020 International Conferences on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*, pp. 735–742, Rhodes, Greece, November 2020.