Hindawi Publishing Corporation
Journal of Sensors
Volume 2015, Article ID 192194, 16 pages
http://dx.doi.org/10.1155/2015/192194

Research Article

A Novel Bioinspired Multiobjective Optimization Algorithm for Designing Wireless Sensor Networks in the Internet of Things

Jun Huang,[1] Liqian Xu,[1] Cong-cong Xing,[2] and Qiang Duan[3]

[1]*School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China*
[2]*Math and Computer Science Department, Nicholls State University, Thibodaux, LA 70310, USA*
[3]*Information Science and Technology Department, The Pennsylvania State University, Abington, PA 19001, USA*

Correspondence should be addressed to Jun Huang; xiaoniuadmin@gmail.com

Received 6 November 2014; Accepted 15 December 2014

Academic Editor: Anand Paul

The design of wireless sensor networks (WSNs) in the Internet of Things (IoT) faces many new challenges that must be addressed through an optimization of multiple design objectives. Therefore, multiobjective optimization is an important research topic in this field. In this paper, we develop a new efficient multiobjective optimization algorithm based on the chaotic ant swarm (CAS). Unlike the ant colony optimization (ACO) algorithm, CAS takes advantage of both the chaotic behavior of a single ant and the self-organization behavior of the ant colony. We first describe the CAS and its nonlinear dynamic model and then extend it to a multiobjective optimizer. Specifically, we first adopt the concepts of "nondominated sorting" and "crowding distance" to allow the algorithm to obtain the true or near optimum. Next, we redefine the rule of "neighbor" selection for each individual (ant) to enable the algorithm to converge and to distribute the solutions evenly. Also, we collect the current best individuals within each generation and employ the "archive-based" approach to expedite the convergence of the algorithm. The numerical experiments show that the proposed algorithm outperforms two leading algorithms on most well-known test instances in terms of Generational Distance, Error Ratio, and Spacing.

1. Introduction

The Internet of Things (IoT) is an emerging paradigm for information collection, communications, process, and application that is rapidly gaining ground in a wide variety of fields, including manufacture, transportation, healthcare, environment surveillance, and many other areas. The basic idea of this new networking and computing paradigm is the pervasive presence of a variety of objects (things), such as sensors, actuators, mobile devices, and RFID tags, which are able to interact with each other and communicate with the Internet infrastructure. Wireless sensor networks (WSNs) form an indispensable ingredient of the IoT for data collections and transmissions, thus having a significant impact on the overall service performance of the IoT. Therefore, optimization of the WSN design plays a crucial role in constructing a high performance IoT for meeting the requirements of various applications.

Designing a large scale WSN for supporting the IoT faces many challenges. These challenges are mainly caused by the limited resources available in WSNs, including battery lifetime, computing capacity, and memory space at sensor nodes, and also by dynamic network topology especially in ad hoc sensor networks. On the other hand, due to the wide spectrum of applications that may be supported by the IoT in the future, WSN design faces various design goals that often conflict with each other, for example, short delay, high throughput, minimal energy consumption, and low cost. Network design in such application scenarios must consider multiple factors in order to achieve trade-offs among multiple objectives to achieve optimal overperformance for the entire network. Therefore, multiobjective optimization algorithms (MOAs) are fundamentally important to WSN design in the IoT development.

MOAs may be employed to address various key problems in WSN design. For example, the sensor node placement

problem involves tradeoffs between multiple objectives such as maximum coverage and minimum energy consumption. When sensor nodes are organized in clusters, the selection of cluster heads is also a challenging issue that must be tackled with a multiobjective optimization vision. Routing strategy is another fairly important problem in WSN design [1], which may have a direct impact on multiple aspects of network performance, including data transmission delay, throughput, and lifetime of individual nodes as well as the entire network. Mobile agents are often used in WSNs to visit a sequence of sensors and fuse impotent data. Optimal agent routes also often need to meet multiple objectives such as minimizing total path delay, loss, and energy consumption as well [2]. These multiobjective problems are all challenging problems that need complex algorithms. On the other hand, special features of WSNs, such as limited computing power, storage space, and battery capacity in particular, bring in new requirements on time and space efficiency of the MOAs that can be applied in such an environment. This motivates research on devising new algorithms for multiobjective optimizations.

During the last twenty years, many MOAs have been proposed [3, 4]. In particular, the multiobjective evolutionary algorithms have been extensively studied [5–14]. For instance, the nondominated sorting genetic algorithm II (NSGA-II) [12] and the strength Pareto evolutionary algorithm 2 (SPEA2) [10] are the most favorable ones used in the field of engineering, economics, and business management [15–17]. On the other hand, some other MOAs have also been proposed with the development of bioinspired heuristics [18–21]. The representative one in this line of research is the multiobjective particle swarm optimization (MOPSO), which is a multiobjective optimizer extended by particle swarm optimization (PSO). In addition, Coello et al. [21] showed that MOPSO can achieve a better performance than both NSGA-II and SPEA2 with some well-known testing instances.

Inspired by a biological experiment of ants' chaotic behavior, a chaotic ant swarm (CAS) optimization algorithm that models ants' chaotic behaviors mathematically has been given in [22]. Specifically, CAS is a derivative-free method; the individuals of the ant colony exchange information and benefit from either their own experiences or the experiences of other individuals, while exploring promising areas of the search space. The model and mechanism of CAS are different from those of ant colony optimization (ACO) [23]. The CAS algorithm has been successfully applied to various areas such as fuzzy system identification [24], economic dispatch [25], computation of multiple global optima [26], data clustering, and parameter identification [27].

In this paper, we mainly focus on extending the CAS algorithm to a multiobjective optimizer. The major contributions of this paper are summarized as follows.

(i) We employ the basic concepts of "nondominated sorting" and "crowding distance" referred to by NSGA-II to allow the proposed algorithm to obtain the true or near Pareto optima.

(ii) We redefine the rule of "neighbor" selection for each individual (ant) to enable the algorithm to converge and to distribute the solutions evenly. Also, we allow CAS to collect the current best individuals within each generation and we employ the "archive-based" approach to speed up the convergence of the algorithm.

(iii) We conduct thorough comparisons between CAS and two leading multiobjective optimization algorithms (MOPSO and NSGA-II) over representative benchmarks. The results show that the proposed algorithm outperforms MOPSO and NSGA-II on most of testing instances in terms of *Generational Distance, Error Ratio*, and *Spacing*.

The remainder of this paper is organized as follows. Section 2 presents notations and basic concepts of multiobjective optimization. Section 3 briefly describes the CAS algorithm. The multiobjective version of CAS and its related analysis are proposed in Section 4. Section 5 shows the experimental results and compares the average performance of the proposed algorithm against that of both MOPSO and NSGA-II with benchmark testing problems. Section 6 draws the conclusion of this paper.

2. Notations and Basic Concepts

In this section, we introduce the basic concepts of multiobjective optimization. Terminology is consistent with that in [12, 13, 21].

Definition 1 (global minimum). Given a function $f : \Omega \subseteq R^n \rightarrow R$, $\Omega \neq \Theta$ for $\vec{x} \in \Omega$ the value $f^* \equiv f(\overrightarrow{x^*}) > -\infty$ is called global minimum if and only if

$$\forall \vec{x} \in \Omega : f\left(\overrightarrow{x^*}\right) \leq f(\vec{x}). \tag{1}$$

In this case, $\overrightarrow{x^*}$ is the minimum solution, f is the objective function, the set Ω is the feasible region ($\Omega \in S$), Θ is the empty set, and S represents the whole search space.

Definition 2 (multiobjective optimization problem). The multiobjective optimization problem is to minimize $F(\vec{x}) = (f_1(\vec{x}), \ldots, f_m(\vec{x}))$ subject to $\vec{x} \in \Omega$, where Ω is the decision (variable) space, R^m is the objective space, and $F : \Omega \rightarrow R^m$ consists of m real-valued objective functions.

Definition 3 (dominate, Pareto set, Pareto front). Let $u = (u_1, \ldots, u_m)$, $v = (v_1, \ldots, v_m) \in R^m$ be two vectors; u is said to *dominate* v if $u_i \leq v_i$ for all $i = 1, \ldots, m$, and $u \neq v$. A point $\overrightarrow{x^*} \in \Omega$ is called (globally) Pareto optimal if there is no $\vec{x} \in \Omega$ such that $F(\vec{x})$ dominates $F(\overrightarrow{x^*})$. The set of all Pareto optimal points, denoted by PS, is called the *Pareto set*. The set of all Pareto objective vectors, PF $= \{F(\vec{x}) \in R^m, \vec{x} \in \text{PS}\}$, is called the *Pareto front*.

3. Chaotic Ant Swarm Algorithm

The phenomenon and behaviors of social insect societies such as ant colonies have fascinated many researchers in recent

years. What particularly strikes the occasional observers as well as the scientists is the high degree of societal organization of ants; that is, the ant colonies can accomplish complex tasks in spite of the limited capability of individual ants. Through information exchange, the global self-organization behavior of social ants emerges. Such emergent behavior was reported in Leptothorax ant colonies which was related to the chaotic dynamics. Global oscillations of colony activity were also reported together with the observation that individual ant behavior can be characterized by low-dimensional strange attractors [28]. Meanwhile, the study of ant chaotic behaviors and of their self-organizing capacities has greatly interested computer scientists in developing models for distributed organizations which are useful in solving difficult optimization and distributed control problems. In the following, we will give the detailed chaotic ant swarm algorithm based on some biological observations and investigations on the chaotic and self-organizing behaviors of ants.

3.1. Chaotic and Self-Organizing Behaviors of Ants. The chaotic behavior of insect was first presented by Cole in his experimental studies on activity cycles of Leptothorax ants [28]. Cole used a solid-state automatic digital camera to examine the dynamical behaviors of isolated individual ants and ant colony. He finally found some interesting results, "the ant behavior may be chaotic. The attractor of the movement activity of single, isolated Leptothorax longispinosus ants has a small, non-integer dimension characteristic of low-dimensional chaos. The activity of entire colonies of ants yields an integer dimension that is consistent with periodicity in activity." He even speculated that, "The existence of chaos animal behavior can have several important implications. Variation in the temporal component of individual behavior may not be due simply to chance variations in the stochastic world, but to deterministic processes that depend on initial conditions." Inspired by Cole's observation, we give a nonlinear dynamic model called CAS to describe the chaotic and self-organizing behaviors of ants.

3.2. Nonlinear Dynamic Model of Chaotic Ant Swarm. ACO algorithms as well as CAS algorithm are both inspired by ant behaviors, but the fundamental principle of these two algorithms are quite different. ACO algorithms, which are based on probability theory, explain how ants find the shortest paths between food sources and their nests by disseminating pheromone. However, ACO algorithms do not consider any chaotic behaviors of single ants. Chaotic ant swarm algorithm, on the other hand, is based on chaotic search strategy and self-organizing of ant colony. We now give the nonlinear dynamic model of CAS as follows.

Consider an ant colony including n ants, which located in the decision (search) space Ω and they try to minimize a function $F : \Omega \rightarrow R^m$. The position of each ant i is denoted by $s_i = (z_{i1}, \ldots, z_{iL})$, where $i = 1, 2, \ldots, n$.

In the initial stage, we employ the chaotic system $z(t+1) = z(t)e^{[3-\psi z(t)]}$ in our equation to perform chaotic search. The system obtained from $z(t+1) = z(t)e^{\mu[1-z(t)]}$ is described by Sole et al. in [29] (for more details please see our previous work [22–28, 30]). Each ant is influenced by its current position, the best position so far, its neighbors and the organization process of the swarm. The individual ant's chaotic behavior is adjusted by introducing a successive decrement of the organization variable $y_i(t)$. Such an organization variable can lead the individual to moving to the new site then acquiring the best fitness. In addition, to achieve the goals of the information exchange between individuals and the movements to the new site aiming on the best fitness, we further introduce an quantity $[p_{id}(t-1) - z_{id}(t-1)]$. The term p_{id} is selected based on the fitness theory which has been substantially developed in optimization theory. Thus, we obtain the following detailed dynamical optimization system of the chaotic ant swarm:

$$\begin{aligned} y_i(t) &= y_i(t-1)^{(1+r_i)}, \\ z_{id}(t) &= z_{id}(t-1)\, e^{[1-e^{-ay_i(t)}][3-\psi_d z_{id}(t-1)]} \\ &\quad + \left[p_{id}(t-1) - z_{id}(t-1)\right] e^{-2ay_i(t)+b}, \end{aligned} \tag{2}$$

where a is a sufficiently large positive constant and can be selected as $a = 200$, b is a constant and $0 \le b \le 2/3$, ψ_d determines the selection of the search range of the dth element of the variable in search space, $r_i \ge 0$ is termed as the organization factor of ant i, $y_i(0) = 0.999$, and $z_{id}(t)$ is the current state of the dth dimension of the individual ant i, where $d = 1, 2, \ldots, L$. $p_{id}(t-1)$ is the best position found by the ith ant and its neighbors within $t-1$ steps, $y_i(t)$ is the current state of the organization variable, t means the current time step, and $t-1$ is the previous step.

$y_i(t)$ and r_i control the convergence of (2); $y_i(t)$ may converge faster if r_i gets larger. The organization factor $r_i = 0$ means the ant swarm is not organized. Under this condition, $e^{-ay_i(t)}$ and $e^{-2ay_i(t)+b}$ are approximately equal to zero, and (2) can be rewritten as $z_{id}(t) = z_{id}(t-1)e^{[3-\psi_d z_{id}(t-1)]}$. If r_i is very large, then the total time of "chaotic" search is small and the system converges very quickly; however, the desired optima (or near optima) cannot be obtained in this case. On the other hand, if r_i is small, then the time of chaotic search is large, the system converges slowly, and the running time will be longer. The value of r_i is thus chosen between $0 \le r_i \le 0.5$. To denote a different r_i of each ant, for example, we can set $r_i = 0.1 + 0.2\,\mathrm{rand}(\,)$, where rand( ) is a uniformly distributed random number in the interval $[0, 1]$. After the chaotic search, $y_i(t)$ approximately equals zero and the convergence of (2) will be mainly determined by $z_{id}(t) = z_{id}(t-1) + \exp(b)[p_{id}(t-1) - z_{id}(t-1)]$. Under this condition, when $0 < b < \ln(2)$, the state $z_{id}(t)$ of (2) will converge to $p_{id}(t)$.

We found that the above model of chaotic ant swarm equation (2) searches for optima in constrained positive or negative intervals. Specifically, if $\psi_d > 0$, (2) can be used to realize the search process in the intervals in which all $z_{id}(t) \ge 0$, and if $\psi_d < 0$, (2) can be used to realize the search process

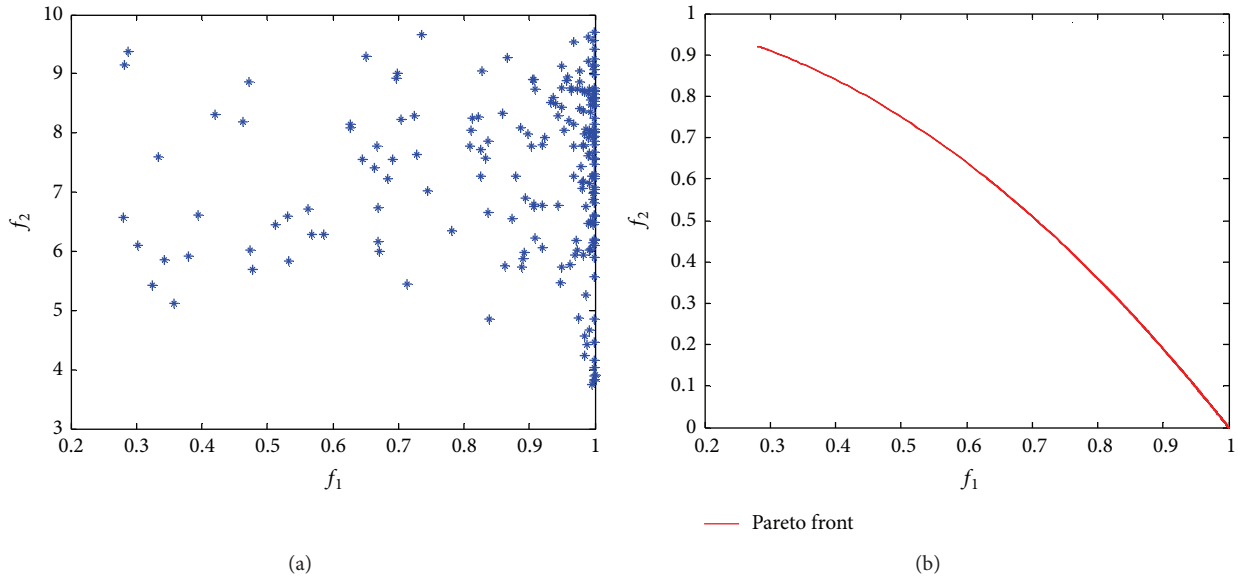

Figure 1: Solutions (a) generated by the algorithm and the Pareto front (b) for problem ZDT6.

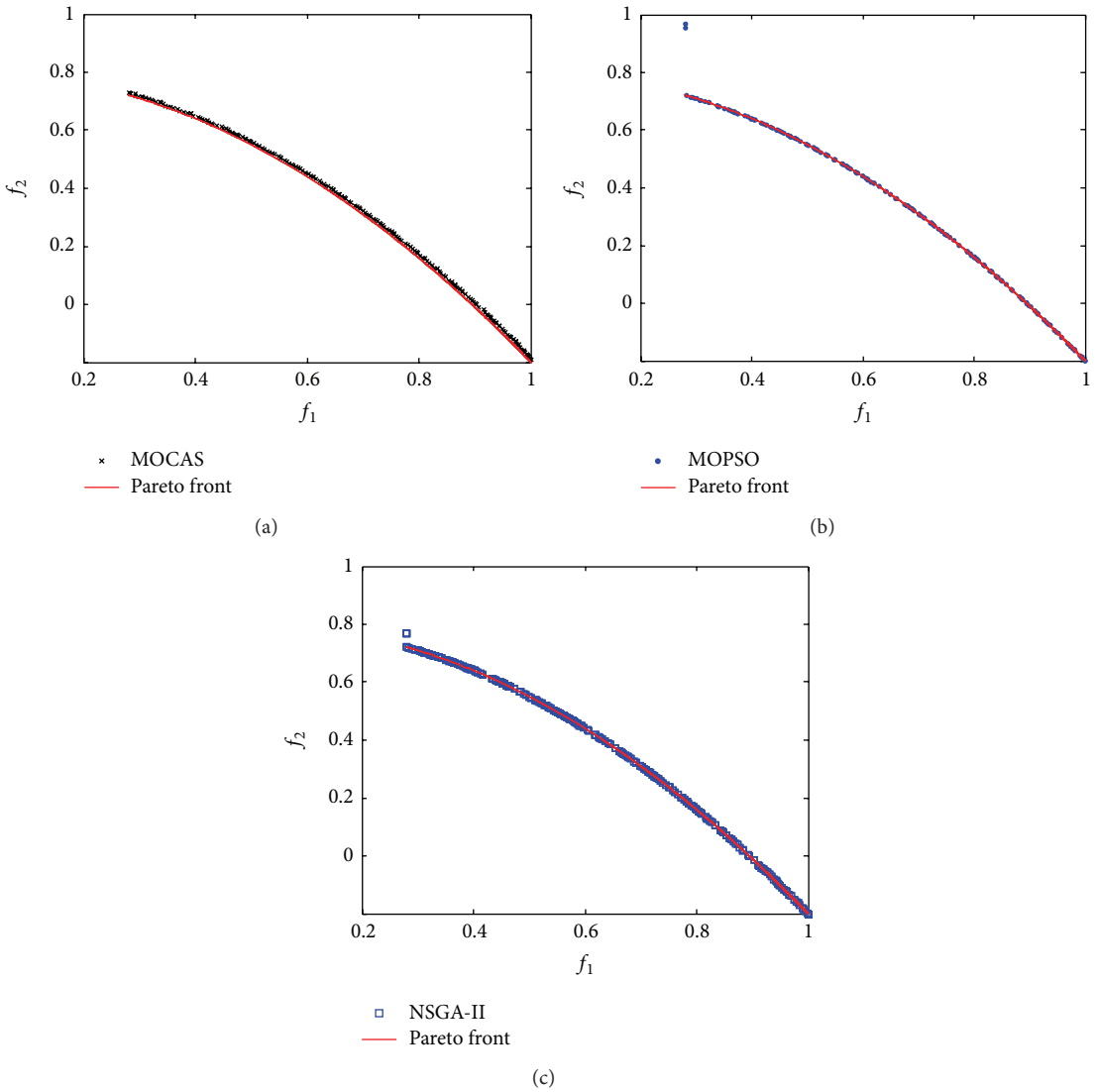

Figure 2: Pareto front generated by the MOCAS (a), MOPSO (b), and NSGA-II (c) for P1.

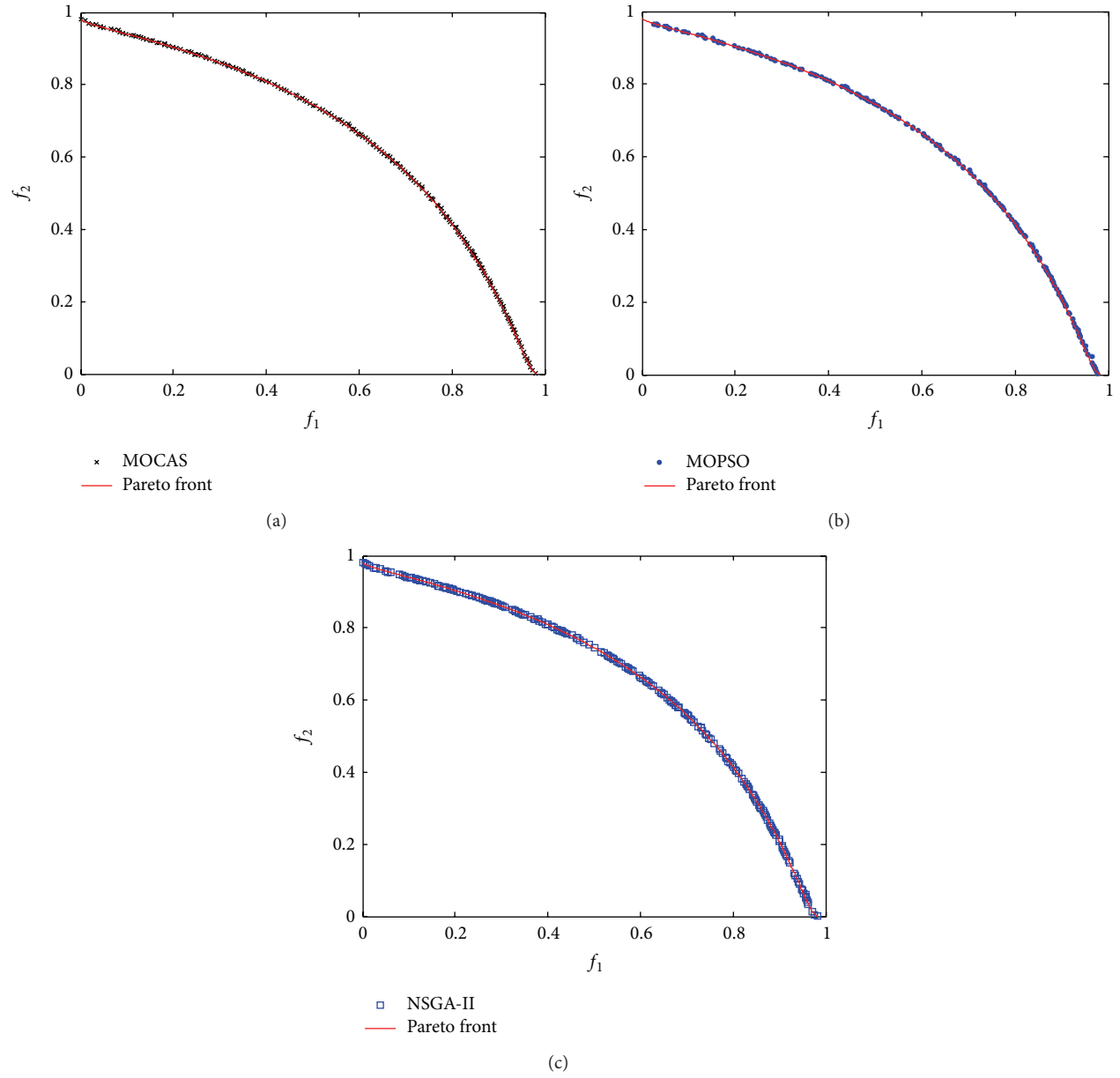

Figure 3: Pareto front generated by the MOCAS (a), MOPSO (b), and NSGA-II (c) for P2.

in the intervals for all $z_{id}(t) \leq 0$. We therefore introduce $V_i(7.5/\psi_d)$ and give the following version of CAS model:

$$y_i(t) = y_i(t-1)^{(1+r_i)},$$

$$z_{id}(t) = \left[z_{id}(t-1) + V_i\frac{7.5}{\psi_d}\right] e^{[1-e^{-ay_i(t)}][3-\psi_d z_{id}(t-1)]} + \left[p_{id}(t-1) - z_{id}(t-1)\right] e^{-2ay_i(t)+b} - V_i\frac{7.5}{\psi_d}, \quad (3)$$

where $0 \leq V_i \leq 1$ determines the search region of ant i and offers the advantage that ants could search diverse regions of the problem space. If $V_i = 1/2$, it means that the chaotic attractor of ant i moves a half to the negative orientation compared to the chaotic attractor of (2). The values of V_i should be suitably selected according to the actual optimization problems. We call (3) the general algorithmic model of chaotic ant swarm. In this model we can select the initial position of an individual ant as $z_{id}(0) = 7.5/\psi_d(1 - V_i)\,\mathrm{rand}(\,)$, where $\psi_d > 0$. More details about CAS are given in [22].

4. Multiobjective Chaotic Ant Swarm Algorithm

4.1. From Single-Objective to Multiobjective. Equation (3) enables the CAS algorithm to obtain the optima or near optima with "satisfactory" precision when handling single-objective optimization problems [22–28, 30]. In a naïve

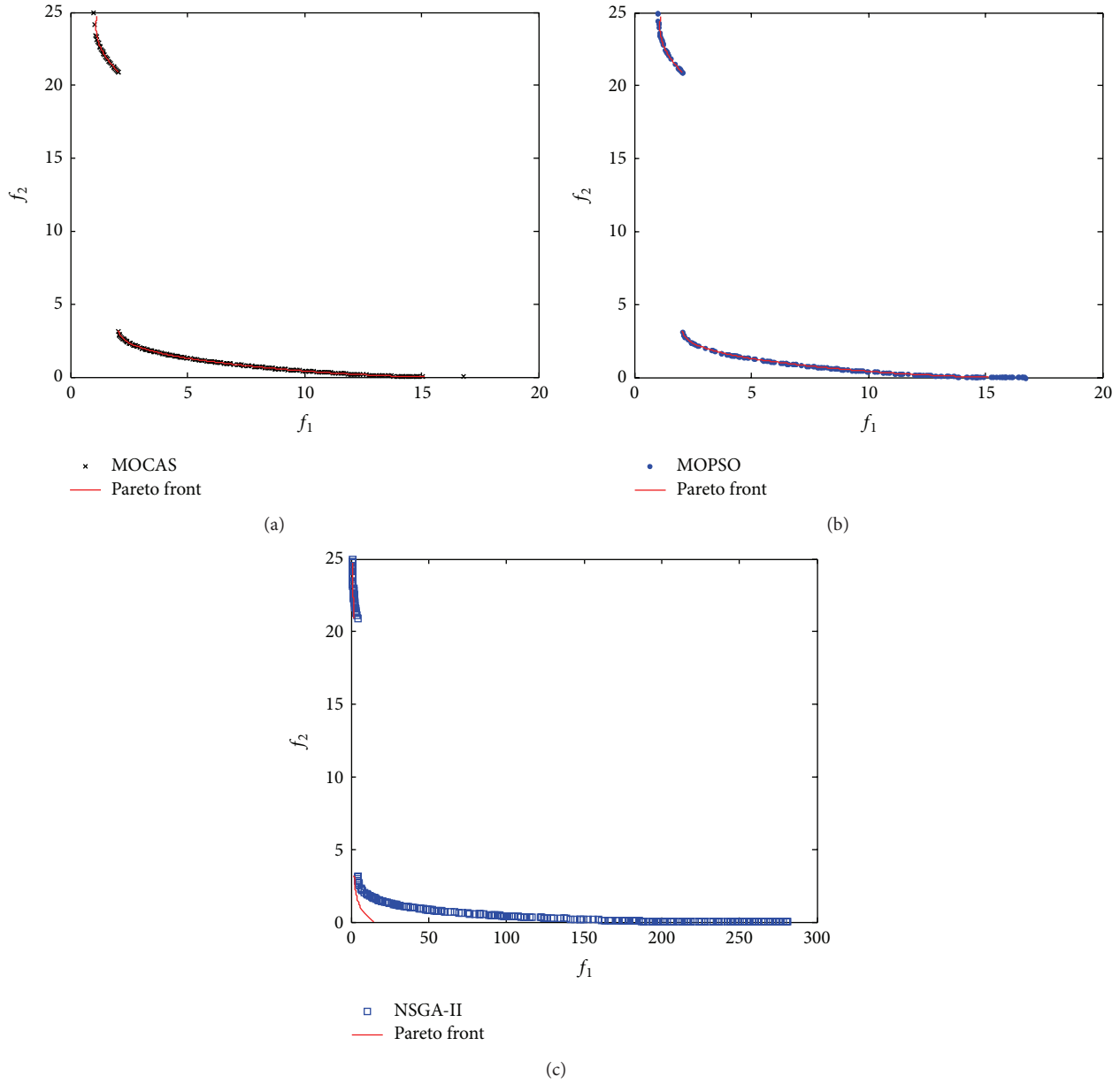

Figure 4: Pareto front generated by the MOCAS (a), MOPSO (b), and NSGA-II (c) for P3.

way, we directly extend the CAS to multiobjective CAS by employing the basic concepts of "nondominated sorting" and "crowding distance" [12]. Unfortunately, the extended CAS algorithm cannot achieve our desired goal. Figure 1 presents the obtained result of such an algorithm when solving a well-known multiobjective optimization problem ZDT6 [12]:

$$f_1(x) = 1 - \exp(-4x_1)\sin^6(6\pi x_1),$$
$$f_2(x) = g(x)\left[1 - \left(\frac{f_1(x)}{g(x)}\right)^2\right], \tag{4}$$

where

$$g(x) = 1 + 9\left(\sum_{i=2}^{n}\frac{x_i}{n-1}\right)^{0.25} \tag{5}$$

subject to $0 \le x_i \le 1, i = 1, \ldots, 10$.

As shown in Figure 1, the obtained results are far from the Pareto set; that is, this naively extended algorithm fails to explore the optima. This failure motivates us to reconsider the behaviors of both individual ants and ant colony.

Practically, in the real world, an individual ant performs the chaotic search at the initial time. If a pioneer ant has found a better position, the neighboring ants will follow him

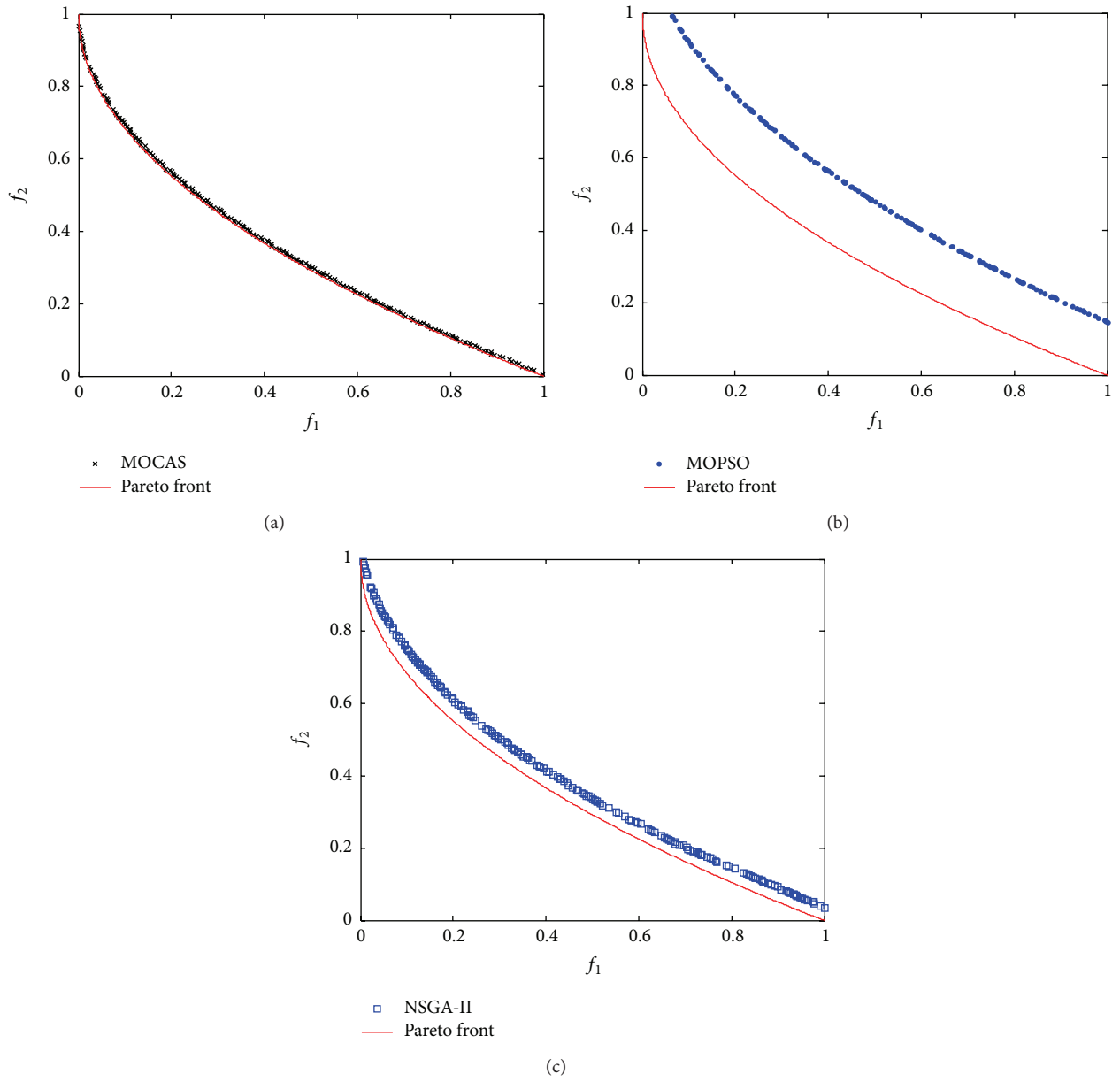

Figure 5: Pareto front generated by the MOCAS (a), MOPSO (b), and NSGA-II (c) for P4.

by exchanging information. With the search being evolved, all of the ants will eventually reach the source of food. This is the basic idea behind the single-objective CAS algorithm. However, the organization of ant colony for multiobjective optimization is quite different. In the single-objective optimization, the algorithm could find the optima based on the assumption that the "better" spatial position of a neighbor leads to a "better" object value. But this assumption does not hold in the multiobjective optimization. The neighbor should be redefined as the individual ant in the current first front to enable the algorithm to converge. On the other hand, individual ants that are in the current first front would travel to each other via information exchange. For example, if ant i and ant j are two neighboring ants in the current front, then ant i would update its position to that of ant j by exchanging information with ant j since ant i supposes ant j is much better than itself. In the same way, ant j will update its position to that of ant i.

4.2. Multiobjective Chaotic Ant Swarm (MOCAS) Algorithm. Besides the differences mentioned above, the archive-based approach in the MOCAS is also employed to store nondominated solutions generated in the past. The use of redefined concept of neighbor selection and the historical record of previously found nondominated vectors allow the MOCAS algorithm to converge to the globally nondominated solutions with fast speed.

The detailed steps of the MOCAS algorithm are given in Algorithm 1.

Step 1. Initialize the main population (ant colony) P;
  for each individual in P
    initialize its position, search range, organization factor and other variables;
  initialize the archive $S = \Phi$;
Step 2. Evaluate each individual in the population;
Step 3. Classify the individuals in P into dominated individuals D and non-dominated individuals ND;
  for each individual p in P
    flag = 0;
    for each individual q in P
      if p is dominated by q
        $D = D \cup \{q\}$;
      else
        $D = D \cup \{p\}$;
        flag = 1;
    if flag == 0
      ND = ND $\cup \{p\}$;
Step 4. Calculate crowding distance for ND;
  Initialize the distance to be zero for all individuals in ND (nd denotes the size of ND);
  for each objective m
    sort the individuals in ND based on objective m;
  assign infinite distance to boundary values for each individual in ND, i.e. ND$(d_1) = \infty$ and ND$(d_{nd}) = \infty$;
  for $i = 2$ to $(nd - 1)$
    $$\mathrm{ND}(d_i) = \mathrm{ND}(d_i) + \frac{\mathrm{ND}(i+1) \cdot m - \mathrm{ND}(i-1) \cdot m}{f_m^{\max} - f_m^{\min}};$$
    /*ND $(i) \cdot m$ is the value of the mth objective function of the ith individual in P*/
Step 5. Update the archive S;
  $S' = S \cup$ ND;
  classify S' into dominated individuals D' and non-dominated individuals ND$'$;
  $S =$ ND$'$;
Step 6. Generate new population by (3) and re-defined concept of neighbor selection;
Step 7. if terminate is true
  Output the population;
else
  goto *step 2*;

Algorithm 1: MOCAS.

Another desired goal of MOCAS is to distribute the obtained solutions on the Pareto front evenly. In the proposed MOCAS, we use the crowding distance mechanism to guide the algorithm to generate the well-distributed Pareto front. Specifically, in Step 6, the algorithm first sorts the ND according to the crowding distance. When the individual with the largest value of crowding distance selects the neighbor, he will determinately select the one with the smallest value of crowding distance in ND. This idea is also inspired by the ant behavior in the real world. When ants found several sources of food, the ant in the crowded source would move toward the uncrowded regions. We will validate the effectiveness of this idea in the next section.

Now we analyze the time complexity of one iteration of the entire algorithm. Step 1 initializes the whole algorithm involving ant colony and some related parameters. It takes $O(N \cdot (D + M))$ time, where N is the population size, D is the dimension of decision variables, and M is number of objectives.

Step 2 steps into the main loop and evaluates objectives for each individual; this procedure takes $O(NM)$ time. Step 3 classifies the population and takes $O(MN^2)$ in the worst case. Step 4 calculates the crowding distance for individuals in the current front; this step takes $O(2MN \log(2N))$. Step 5 updates the archive by nondominated sorting, so it has the same time complexity of Step 3 $O(MN^2)$. Step 6 generates a new population by (3); this will take $O(2MN \log(2N))$. Hence, the time complexity of one iteration of MOCAS in the worst case is $O(MN^2)$.

5. Numeric Simulations

In this section, we test the MOCAS algorithm with several well-known problems and we compare its performance against that of the state-of-the-art multiobjective optimization algorithms, NSGA-II (the code is available at http://www.iitk.ac.in/kangal/codes.shtml) and MOPSO (the code is available at http://www.cs.cinvestav.mx/~EVOCINV/software.html). Another important algorithm SPEA2 is not compared here since its performance is worse than that of NSGA-II in most cases [12]. We use the typical parameter settings for these three algorithms except that the population size and the number of generations are all set to 200 and

Table 1: Testing functions used in this study.

Instance	Objective functions	Variable boundaries	Optimal Solutions
P1	$f_1(x) = 1 - e^{-4x_1}\sin^6(6\pi x_1)$ $f_2(x) = g(x)\left(1 - \left(\frac{f_1(x)}{g(x)}\right)^2\right)$ $g(x) = 1 + 9\left(\sum_{i=2}^{6}\frac{x_i}{4}\right)^2$	$[0, 1]$	$x_1 \in [0, 1]$ $x_i = 0$ $i = 2, \ldots, 6$
P2	$f_1(x) = 1 - \exp\left(-\sum_{i=1}^{3} x_i - \left(\frac{1}{\sqrt{3}}\right)^2\right)$ $f_2(x) = 1 - \exp\left(-\sum_{i=1}^{3} x_i + \left(\frac{1}{\sqrt{3}}\right)^2\right)$	$[-4, 4]$	$x_1 = x_2 = x_3 \in \left[-\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}\right]$
P3	$f_1(x) = \left[1 + (A_1 - B_1)^2 + (A_2 - B_2)^2\right]$ $f_2(x) = \left[(x_1 + 3)^2 + (x_2 + 1)^2\right]$ $A_1 = 0.5\sin 1 - 2\cos 1 + \sin 2 - 1.5\cos 2$ $A_2 = 1.5\sin 1 - \cos 1 + 2\sin 2 - 0.5\cos 2$ $B_1 = 0.5\sin x_1 - 2\cos x_1 + \sin x_2 - 1.5\cos x_2$ $B_2 = 1.5\sin x_1 - \cos x_1 + 2\sin x_2 - 0.5\cos x_2$	$[-\pi, \pi]$	$x_1 = -\pi$ $x_2 \in \left[-\frac{4\pi}{3}, \frac{\pi}{4}\right]$
P4	$f_1(x) = x_1$ $f_2(x) = g(x)\left(1 - \sqrt{\frac{x_1}{g(x)}}\right)$ $g(x) = 1 + \frac{9\left(\sum_{i=2}^{n} x_i\right)}{n-1}$	$[0, 1]$ $n = 30$	$x_1 \in [0, 1]$ $x_i = 0$ $i = 2, \ldots, 10$
P5	$f_1(x) = x_1$ $f_2(x) = g(x)\left(1 - \left(\frac{x_1}{g(x)}\right)^2\right)$ $g(x) = 1 + \frac{9\left(\sum_{i=2}^{n} x_i\right)}{n-1}$	$[0, 1]$ $n = 30$	$x_1 \in [0, 1]$ $x_i = 0$ $i = 2, \ldots, 10$
P6	$f_1(x) = x_1$ $f_2(x) = g(x)\left[1 - \sqrt{\frac{x_1}{g(x)}} - \left(\frac{x_1}{g(x)}\right)\sin(10\pi x_1)\right]$ $g(x) = 1 + \frac{9\left(\sum_{i=2}^{n} x_i\right)}{n-1}$	$[0, 1]$ $n = 30$	$x_1 \in [0, 1]$ $x_i = 0$ $i = 2, \ldots, 10$
P7	$f_1(x) = 1 - \exp(-4x_1)\sin^6(6\pi x_1)$ $f_2(x) = g(x)\left[1 - \left(\frac{f_1(x)}{g(x)}\right)^2\right]$ $g(x) = 1 + 9\left(\frac{\sum_{i=2}^{n} x_i}{n-1}\right)^{0.25}$	$[0, 1]$ $n = 10$	$x_1 \in [0, 1]$ $x_i = 0$ $i = 2, \ldots, 10$
P8	$f_1(x) = x^2$ $f_2(x) = (x - 2)^2$	$[-10^3, 10^3]$	$x \in [0, 2]$

2000, respectively. We run each test instance 30 times to statistically evaluate the performance of three algorithms.

5.1. Simulation Results. The testing problems are listed in Table 1, which are the most frequently used testing instances in multiobjective optimizations. All problems have two objective functions. None of these problems have any constraints. Table 1 also describes the number of variables, their boundaries, and the optima solutions for each problem.

Figures 2–9 depict the nondominated solutions produced by the MOCAS, MOPSO, and NSGA-II algorithms, respectively, when they are applied to each of the problems in Table 1. The red curve in these figures represents the true Pareto front for the corresponding testing problem. It can be seen and intuitively observed from these figures that our proposed MOCAS algorithm either (1) clearly outperforms both MOPSO and NSGA-II algorithms, or (2) outperforms one of the two but performs roughly the same with the other, or (3) performs roughly the same with both of them, for all testing cases. This immediately suggests an average superiority of the MOCAS algorithm over the MOPSO and NSGA-II algorithms.

Specifically, Figures 5 and 6 show that the output of the MOCAS algorithm converges to the true Pareto front for testing problems P4 and P5 while neither the MOPSO algorithm nor the NSGA-II algorithm is able to do so for either of the two problems. Figure 4 shows the comparison on a noncontinuous testing function. The results demonstrate

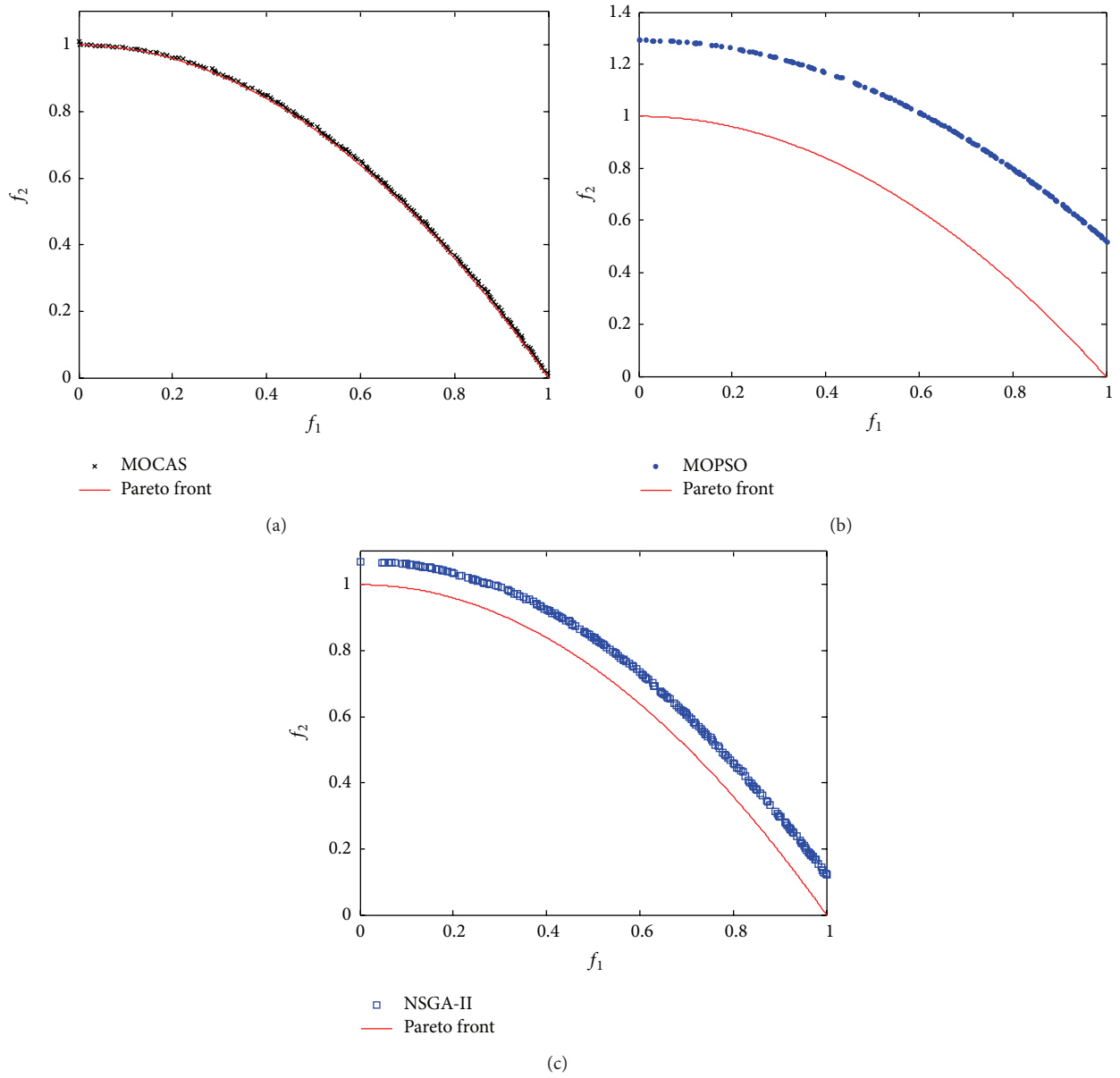

Figure 6: Pareto front generated by the MOCAS (a), MOPSO (b), and NSGA-II (c) for P5.

that the performance of the MOCAS algorithm is roughly equivalent to that of MOPSO but superior to that of NSGA-II for testing problem P3, and Figure 7 demonstrates that the performance of the MOCAS algorithm is roughly equivalent to that of NSGA-II but superior to that of MOPSO for testing problem P6. Figures 2, 3, 8, and 9 all suggest that the performance of the MOCAS algorithm is closely equivalent to that of both MOPSO and NSGA-II for testing problems P1, P2, P7, and P8, respectively. Quantitatively detailed performance analysis is given in the next subsection to show the precise comparison of three algorithms.

5.2. Performance Evaluations. In order to further evaluate the performance of these three algorithms quantitatively, we adopt the following three performance metrics as suggested in [21, 31–33].

5.2.1. Generational Distance (GD). This metric measures how far the obtained nondominated solutions are from the true Pareto front. Its definition is

$$\mathrm{GD} = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n}, \tag{6}$$

where n is the number of obtained nondominated solutions and d_i is the Euclidean distance (measured in objective space) between the ith nondominated solution and the Pareto front.

5.2.2. Error Ratio (ER). This metric was originally proposed by Van Veldhuizen [32] to measure the percentage of obtained nondominated solutions that are not on the Pareto front:

$$\mathrm{ER} = \frac{\sum_{i=1}^{n} e_i}{n}, \tag{7}$$

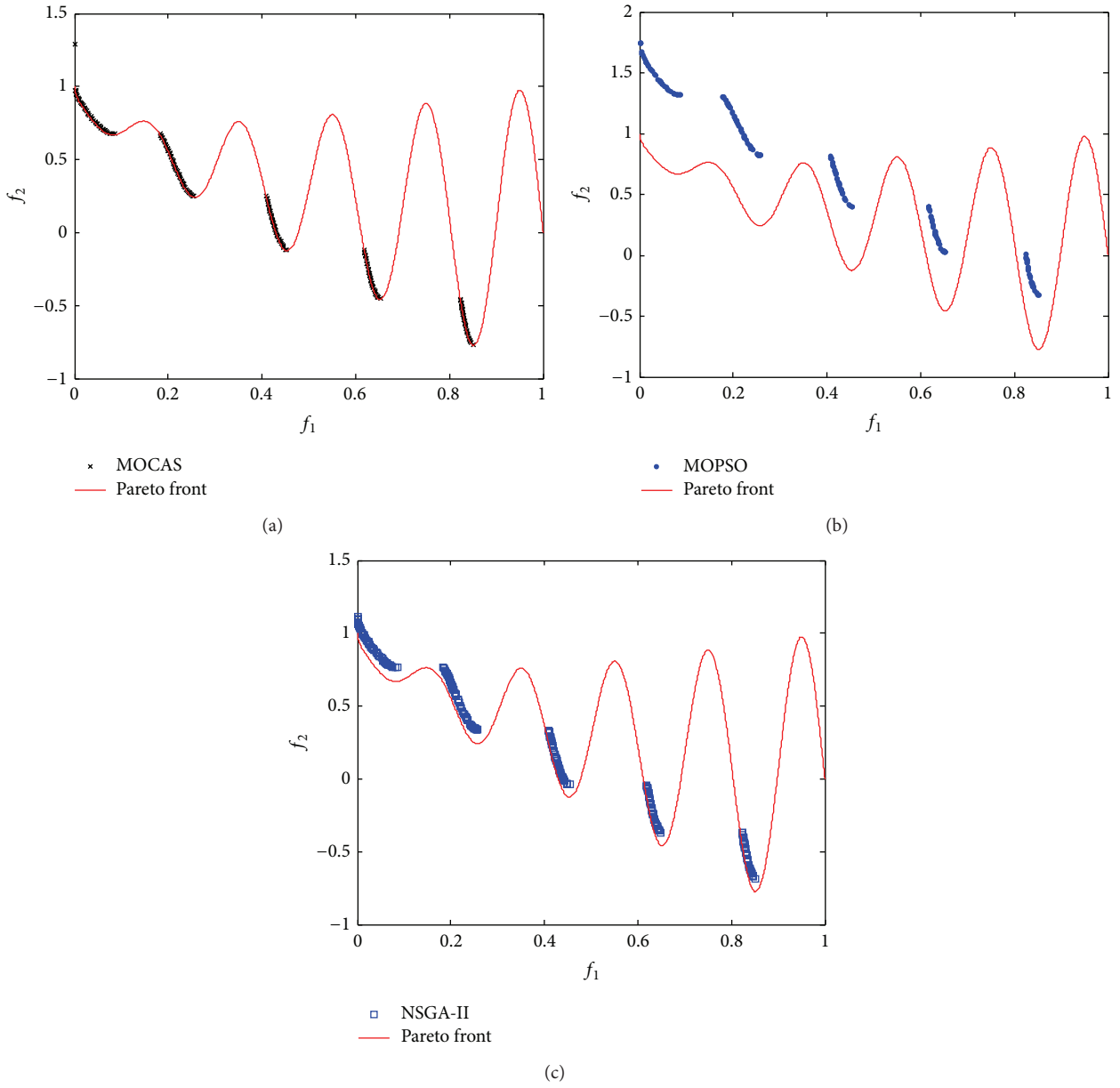

Figure 7: Pareto front generated by the MOCAS (a), MOPSO (b), and NSGA-II (c) for P6.

where n is the number of obtained nondominated solutions. $e_i = 0$ means the ith solution is a member of the Pareto set, and $e_i = 1$, otherwise.

5.2.3. Spacing (SP). This metric measures the extent of spread of obtained nondominated solutions. It is defined as

$$\mathrm{SP} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(\overline{d} - d_i\right)^2}, \tag{8}$$

where $d_i = \min_j[|f_1^i(\vec{x}) - f_1^j(\vec{x})| + |f_2^i(\vec{x}) - f_2^j(\vec{x})|]$, $i, j = 1, \ldots, n$, $\overline{d}$ is the mean of all d_i's, and n is the number of obtained nondominated solutions.

It should be noted that metrics GD and ER describe the first goal of multiobjective optimization algorithms as mentioned previously in the design of MOCAS, while SP reflects the second goal. In the following, we present the comparison for three algorithms with respect to above performance metrics.

Tables 2, 3, 4, 5, 6, 7, 8, and 9 show the comparison of the results from the three algorithms with respect to GD, ER, and SP, where the boldfaced numbers indicate the best average CD (or ER or SP) across the three algorithms. It can be seen that the average performance of MOCAS is the best when the three algorithms run the test problems P2, P3, P4, P5, P6, and P8. For testing problems P1 and P7, however, the tables show that the NSGA-II has the best GD and ER, but the MOCAS has the best SP. By zooming in

Table 2: Comparison results of three metrics for P1.

	MOCAS			MOPSO			NSGA-II		
	GD	ER	SP	GD	ER	SP	GD	ER	SP
Best	0.003346	0.915	0.006114	0.021315	0.03	0.045803	0.006018	0.025	$2.03E-07$
Worst	0.057707	0.995	0.186609	0.054298	0.206522	0.815469	0.060076	0.09	0.342528
Average	0.024415	0.976	**0.082727**	0.034578	0.069526	0.344084	**0.023784**	**0.054**	0.094153
Median	0.022859	0.985	0.086083	0.035318	0.055	0.322122	0.021513	0.045	0.044569
Std. dev.	0.019502	0.02481	0.060023	0.009498	0.050566	0.275979	0.016115	0.025473	0.118375

Table 3: Comparison results of three metrics for P2.

	MOCAS			MOPSO			NSGA-II		
	GD	ER	SP	GD	ER	SP	GD	ER	SP
Best	$5.66E-05$	0	0.002721	$8.72E-05$	0	$3.55E-03$	$9.32E-05$	0	0.003052
Worst	$7.07E-05$	0.005	0.003334	0.000141	0.025	0.004423	0.000129	0.02	0.003885
Average	**6.45E − 05**	**0.0005**	**0.003032**	0.000123	0.0065	0.003828	0.000107	0.009	0.003314
Median	$6.57E-05$	0	0.003015	0.00013	0.005	0.003812	0.000107	0.0125	0.003282
Std. dev.	$4.66E-06$	0.001581	0.000207	$1.83E-05$	0.007835	$2.62E-04$	$1.05E-05$	0.008097	0.000226

Table 4: Comparison results of three metrics for P3.

	MOCAS			MOPSO			NSGA-II		
	GD	ER	SP	GD	ER	SP	GD	ER	SP
Best	0.008367	0.295	0.056603	0.017705	0.825	0.131108	8.94454	1	0.598082
Worst	0.019111	0.38	0.229375	0.036684	0.86	0.29519	9.73034	1	0.753096
Average	**0.01012**	**0.331429**	**0.110787**	0.025308	0.8405	0.187466	9.34286	1	0.672726
Median	0.00917	0.33	0.064751	0.023319	0.84	0.177388	9.37408	1	0.665563
Std. dev.	0.003223	0.029681	0.079856	0.00636	0.010659	0.049485	0.22547	0	0.055491

Table 5: Comparison results of three metrics for P4.

	MOCAS			MOPSO			NSGA-II		
	GD	ER	SP	GD	ER	SP	GD	ER	SP
Best	0.001762	0.52	0.003063	0.010027	1	0.004461	0.000405	0.985	0.003226
Worst	0.003037	0.75	0.003842	0.016015	1	0.005487	0.015986	1	0.213161
Average	**0.002539**	**0.623**	**0.003379**	0.011864	1	0.005067	0.004408	0.997	0.039783
Median	0.002517	0.6125	0.003244	0.011309	1	0.005196	0.000546	1	0.006887
Std. dev.	0.000372	0.082131	0.000288	0.002002	0	0.000348	0.006263	0.00483	0.068458

Table 6: Comparison results of three metrics for P5.

	MOCAS			MOPSO			NSGA-II		
	GD	ER	SP	GD	ER	SP	GD	ER	SP
Best	0.002383	0.31	$2.18E-10$	0.018673	1	0.00268	0.00032	1	0.003371
Worst	0.004709	0.58	0.005356	0.06841	1	0.313675	0.028573	1	0.004965
Average	**0.003966**	**0.4565**	**0.003019**	0.035906	1	0.064442	0.005302	1	0.004118
Median	0.004256	0.455	0.003241	0.032347	1	0.014006	0.001079	1	0.004122
Std. dev.	0.000718	0.082329	0.001462	0.016319	0	0.113717	0.009635	0	0.00042

Table 7: Comparison results of three metrics for P6.

	MOCAS			MOPSO			NSGA-II		
	GD	ER	SP	GD	ER	SP	GD	ER	SP
Best	0.000875	0.01	0.00271	0.018633	1	0.003987	0.000275	0.905	0.005761
Worst	0.001535	0.035	0.003566	0.031005	1	0.210964	0.015983	1	0.006931
Average	**0.001174**	**0.022**	**0.003229**	0.024214	1	0.060328	0.00436	0.97	0.006255
Median	0.001165	0.02	0.003323	0.024216	1	0.022439	0.001551	0.9925	0.006259
Std. dev.	0.00022	0.008563	0.000248	0.004111	0	0.078422	0.005767	0.037565	0.000365

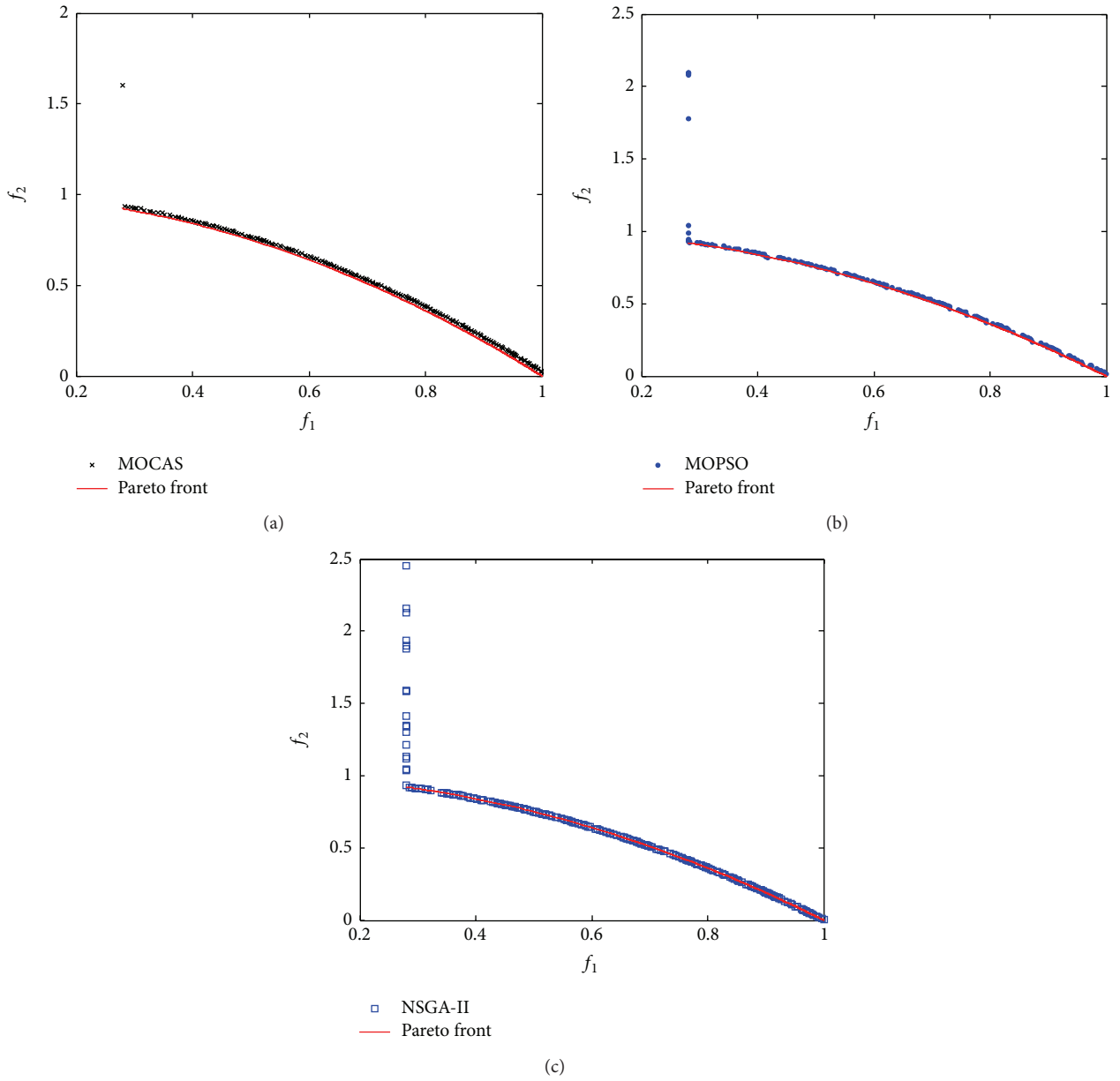

Figure 8: Pareto front generated by the MOCAS (a), MOPSO (b), and NSGA-II (c) for P7.

Table 8: Comparison results of three metrics for P7.

	MOCAS			MOPSO			NSGA-II		
	GD	ER	SP	GD	ER	SP	GD	ER	SP
Best	0.013159	1	0.002741	0.012502	0.636364	0.038502	0.003501	0.04	0.004511
Worst	0.03909	1	0.197246	0.343949	1	0.451114	0.053784	0.11	0.320704
Average	0.027409	1	**0.104**	0.211105	0.849299	0.16362	**0.016799**	**0.072**	0.135646
Median	0.031508	1	0.09552	0.230836	0.908974	0.123775	0.009905	0.0725	0.095691
Std. dev.	0.009926	0	0.071008	0.128875	0.128953	0.129505	0.015584	0.02429	0.135664

the corresponding figures (Figures 2 and 8) and inspecting the details, we see that the MOCAS would not converge to the true Pareto front, but it has a fairly good distribution of obtained solutions. This observation matches the quantitative performance evaluations.

6. Conclusion

We proposed in this paper a novel bioinspired multiobjective optimization algorithm named MOCAS for designing wireless sensor networks in the Internet of Things by extending

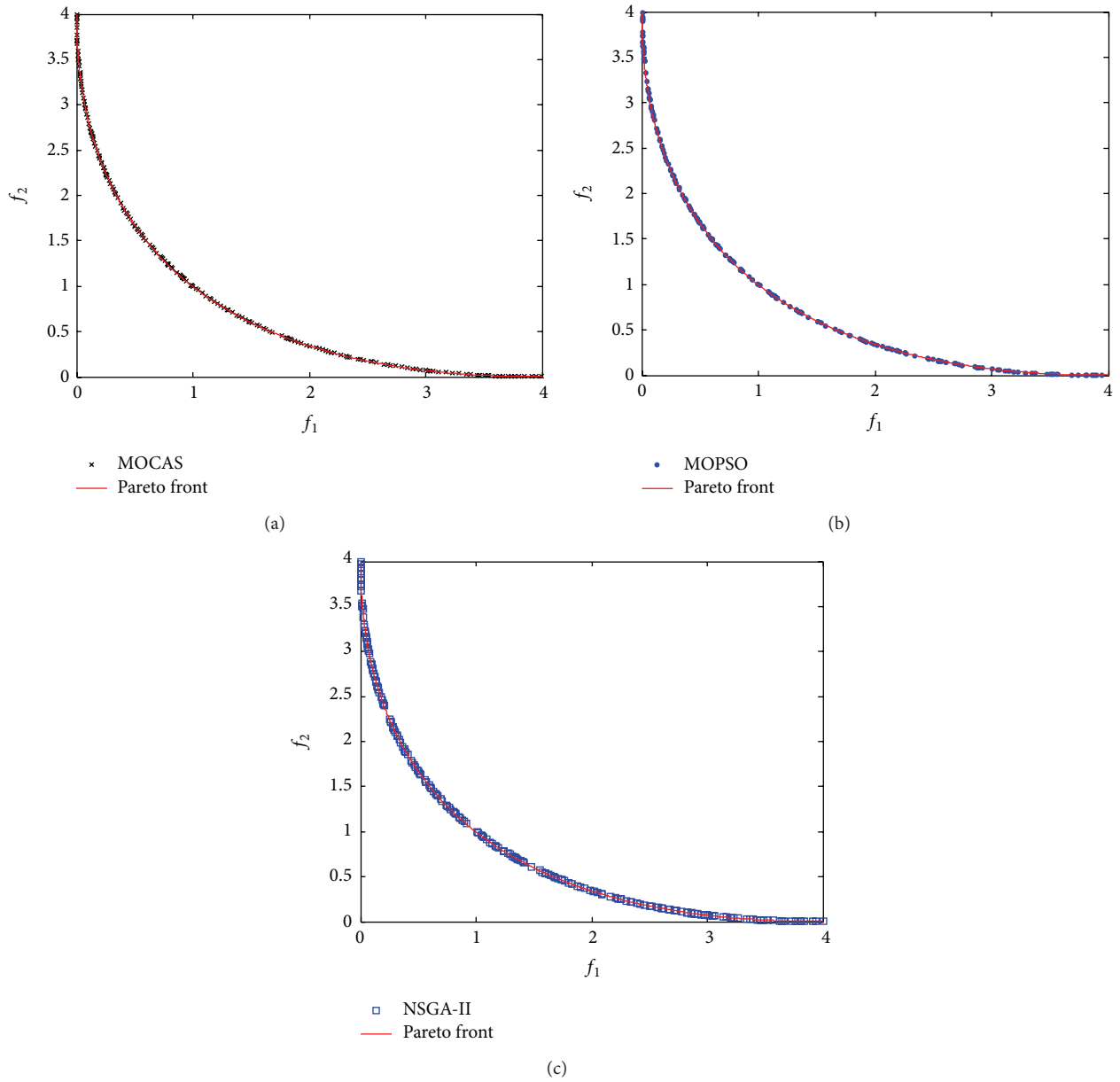

Figure 9: Pareto front generated by the MOCAS (a), MOPSO (b), and NSGA-II (c) for P8.

Table 9: Comparison results of three metrics for P8.

	MOCAS			MOPSO			NSGA-II		
	GD	ER	SP	GD	ER	SP	GD	ER	SP
Best	0	0	0.012175	0	0	0.017289	0	0	0.016588
Worst	0	0	0.016525	0	0	0.023579	0	0	0.020291
Average	0	0	**0.014496**	0	0	0.020478	0	0	0.017275
Median	0	0	0.014371	0	0	0.020433	0	0	0.01685
Std. dev.	0	0	0.001377	0	0	0.001774	0	0	0.001093

previous work on chaotic ant swarms. A straightforward extension from the single-objective optimization CAS model to the multiobjective optimization CAS model was initially attempted by introducing the "nondominated sorting" and "crowding distance" notions into the single-objective optimization CAS model. This approach however turns out to be a failure since its outcomes are not even close to the true Pareto front. We then redefined the concept of "neighbors" and "neighbor-selecting" rules and incorporated an "archive-based" approach into the algorithm allowing the resulting MOCAS algorithm to converge fast to the true Pareto front with an evenly distributed set of solutions. By testing MOCAS

on some well-known multiobjective optimization problems and comparing the results with those produced by the state-of-the-art peer algorithms MOPSO and NSGA-II, we have seen that the proposed MOCAS algorithm outperforms the other two algorithms on average, which evidently shows the competitiveness of the MOCAS algorithm in dealing with multiobjective optimization problems.

Since the variables of MOCAS are bounded by symmetric intervals in this paper, one issue that we would like to explore in the future is how to set the parameter for the algorithm to cover a wider range of multiobjective optimization problems. Another interesting issue that we also would like to study is the further extension of MOCAS to handle constrained multiobjective optimization problems.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] S. Okdem and D. Karaboga, "Routing in wireless sensor networks using ant colony optimization," in *Proceedings of the 1st NASA/ESA Conference on Adaptive Hardware and Systems (AHS '06)*, pp. 401–404, 2006.

[2] T. Camilo, C. Carreto, J. S. Silva et al., "An energy-efficient ant-based routing algorithm for wireless sensor networks," in *Ant Colony Optimization and Swarm Intelligence*, pp. 49–59, Springer, Berlin, Germany, 2006.

[3] F. Y. Cheng and X. S. Li, "Generalized center method for multiobjective engineering optimization," *Engineering Optimization*, vol. 31, no. 5, pp. 641–661, 1999.

[4] T. Marler, *Multi-Objective Optimization: Concepts and Methods for Engineering*, VDM Publishing, Saarbrücken, Germany, 2009.

[5] M. Fonseca and P. J. Fleming, "An overview of evolutionary algorithms in multiobjective optimization," *Evolutionary Computation*, vol. 3, no. 1, pp. 1–16, 1995.

[6] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*, John Wiley & Sons, Chichester, UK, 2001.

[7] A. C. Coello, "20 years of evolutionary multi-objective optimization: what has been done and what remains to be done," in *Computational Intelligence: Principles and Practice*, G. Y. Yen and D. B. Fogel, Eds., chapter 4, pp. 73–88, IEEE Computational Intelligence Society, New York, NY, USA, 2006.

[8] J. D. Schaffer, *Some experiments in machine learning using vector evaluated genetic algorithms [Ph.D. dissertation]*, Vanderbilt University, Nashville, Tenn, USA, 1984.

[9] J. D. Knowles and D. W. Corne, "Approximating the nondominated front using the Pareto archived evolution strategy," *Evolutionary computation*, vol. 8, no. 2, pp. 149–172, 2000.

[10] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: improving the strength pareto evolutionary algorithm," Tech. Rep. 103, Swiss Federal Institute of Technology, Zürich, Switzerland, 2001.

[11] A. Coello Coello and G. T. Pulido, "Multiobjective optimization using a micro-genetic algorithm," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '01)*, L. Spector, E. D. Goodman, A. Wu et al., Eds., pp. 274–282, San Francisco, Calif, USA, 2001.

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

[13] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated pareto sets, MOEA/ D and NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 284–302, 2009.

[14] B. Soylu and M. Köksalan, "A favorable weight-based evolutionary algorithm for multiple criteria problems," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 2, pp. 191–205, 2010.

[15] J. Huang and Y. Liu, "MOEAQ: a QoS-aware multicast routing algorithm for MANET," *Expert Systems with Applications*, vol. 37, no. 2, pp. 1391–1399, 2010.

[16] M. A. Abido, "Multiobjective evolutionary algorithms for electric power dispatch problem," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 315–329, 2006.

[17] T. Siegfried, S. Bleuler, M. Laumanns, E. Zitzler, and W. Kinzelbach, "Multiobjective groundwater management using evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 229–242, 2009.

[18] M. P. Hansen, "Tabu search for multiobjective optimization," in *Proceedings of the 13th International Conference Multiple Criteria Decision Making (MCDM '97)*, pp. 1–16, Cape Town, South Africa, 1997.

[19] E. L. Ulungu, P. H. Teghem, and D. Tuyttens, "MOSA method: a tool for solving multiobjective combinatorial optimization problems," *Journal of Multi-Criteria Decision Analysis*, vol. 8, pp. 221–236, 1999.

[20] A. Jaszkiewicz, "On the computational efficiency of multiple objective metaheuristics. The knapsack problem case study," *European Journal of Operational Research*, vol. 158, no. 2, pp. 418–433, 2004.

[21] C. A. C. Coello, G. T. Pulido, and M. S. Lechuga, "Handling multiple objectives with particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 256–279, 2004.

[22] L. Li, Y. Yang, H. Peng, and X. Wang, "An optimization method inspired by "chaotic" ant behavior," *International Journal of Bifurcation and Chaos in Applied Sciences and Engineering*, vol. 16, no. 8, pp. 2351–2364, 2006.

[23] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 26, no. 1, pp. 29–41, 1996.

[24] L. Li, Y. Yang, and H. Peng, "Fuzzy system identification via chaotic ant swarm," *Chaos, Solitons and Fractals*, vol. 41, no. 1, pp. 401–409, 2009.

[25] J. Cai, X. Ma, L. Li, Y. Yang, H. Peng, and X. Wang, "Chaotic ant swarm optimization to economic dispatch," *Electric Power Systems Research*, vol. 77, no. 10, pp. 1373–1380, 2007.

[26] L. Li, Y. Yang, and H. Peng, "Computation of multiple global optima through chaotic ant swarm," *Chaos, Solitons and Fractals*, vol. 40, no. 3, pp. 1399–1407, 2009.

[27] L. Li, Y. Yang, H. Peng, and X. Wang, "Parameters identification of chaotic systems via chaotic ant swarm," *Chaos, Solitons and Fractals*, vol. 28, no. 5, pp. 1204–1211, 2006.

[28] B. J. Cole, "Is animal behaviour chaotic? Evidence from the activity of ants," *Proceedings of the Royal Society B: Biological Sciences*, vol. 244, no. 1311, pp. 253–259, 1991.

[29] R. V. Sole, O. Miramontes, and B. C. Goodwin, "Oscillations and chaos in ant societies," *Journal of Theoretical Biology*, vol. 161, no. 3, pp. 343–357, 1993.

[30] H. Peng, L. Li, Y. Yang, and F. Liu, "Parameter estimation of dynamical systems via a chaotic ant swarm," *Physical Review E: Statistical, Nonlinear, and Soft Matter Physics*, vol. 81, no. 1, Article ID 016207, 2010.

[31] D. A. van Veldhuizen and G. B. Lamont, "Multiobjective evolutionary algorithm research: a history and analysis," Tech. Rep. TR-98-03, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, USA, 1998.

[32] D. A. Van Veldhuizen, *Multiobjective evolutionary algorithms: Classifications, analyzes, and new innovations [Ph.D. thesis]*, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Wright-Patterson Air Force Base, Ohio, ISA, 1999.

[33] J. R. Schott, *Fault tolerant design using single and multicriteria genetic algorithm optimization [M.S. thesis]*, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Mass, USA, May 1995.