Soft Computing (2019) 23:10681–10697
https://doi.org/10.1007/s00500-019-04004-4

FOUNDATIONS

# A hybrid many-objective cuckoo search algorithm

Zhihua Cui[1] · Maoqing Zhang[2] · Hui Wang[3] · Xingjuan Cai[1] · Wensheng Zhang[4]

Published online: 24 April 2019

**Abstract**
Cuckoo search (CS) is an excellent population-based algorithm and has shown promising performance in dealing with single- and multi-objective optimization problems. However, for many-objective optimization problems (MaOPs), CS cannot be directly employed. So far, few paper have been reported to use CS to solve MaOPs. In this paper, we try to propose a hybrid many-objective cuckoo search (HMaOCS) for MaOPs. In HMaOCS, the standard CS is firstly modified to effectively deal with MaOPs. Then, non-dominated sorting and the strategy of reference points are employed to ensure the convergence and diversity. In order to verify the performance of HMaOCS, DTLZ and WFG benchmark sets are utilized in the experiments. Experimental results show that HMaOCS can achieve promising performance compared with five other well-known many-objective optimization algorithms.

**Keywords** Cuckoo search · Many-objective optimization problems · Non-dominated sorting · Reference points

## 1 Introduction

In the real world, there are many complex optimization problems (Abdel-Baset et al. 2018; Cortés et al. 2018; Cui et al. 2017a, b; Wang et al. 2018). Many researchers have contributed great efforts to solve them (Pandey et al. 2018; Shan et al. 2018; Sun et al. 2014; Zhao et al. 2018) in the past decades. In theory, although many efficient methods (Cai et al. 2016, 2018; Cui et al. 2019b; Fan et al. 2018) have been proposed, there are still some intractable problems (Cui et al. 2018; Pooja et al. 2018; Wang et al. 2017; Yigit et al. 2018) whose complexity is beyond current available methods. Multi-objective optimization problems (MOPs) are the problems which have two or three objectives, and they arise in many areas, such as engineering, economics and biology (Coelho et al. 2013; Cui et al. 2019a; Niu et al. 2018). Such optimization problems often have multiple conflicting objectives, which cannot be optimized simultaneously. Thus, no single best solution can be obtained. Instead, a set of trade-off solutions are needed to approximate the true Pareto front. Over the past several years, many efficient and effective algorithms have been proposed to tackle MOPs, such as NSGA-II (Deb et al. 2002b), MOEA/D (Zhang and Li 2007) and SPEA2 (Zitzler et al. 2001). However, in practical applications, there are still many problems with more than three objectives, and these problems are called many-objective optimization problems (MaOPs) (Cui et al. 2019c). MaOPs pose various challenges to the existing multi-objective optimization algorithms. The first one is high proportion of the non-dominated solutions. With increase in objectives, more and more solutions become non-dominated and sometimes the number of solutions is more than the population slots, thus resulting in slowing down the search process. The second

Communicated by A. Di Nola.

✉ Maoqing Zhang
maoqing_zhang@163.com

✉ Xingjuan Cai
xingjuancai@163.com

Zhihua Cui
zhihua.cui@hotmail.com

Hui Wang
huiwang@whu.edu.cn

Wensheng Zhang
wensheng.zhang@ia.ac.cn

1 Complex System and Computational Intelligence Laboratory, Taiyuan University of Science and Technology, Taiyuan 030024, Shanxi, China

2 School of Electronics and Information, Tongji University, Shanghai 201804, China

3 School of Information Engineering, Nanchang Institute of Technology, Nanchang 330099, China

4 Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China

challenge is diversity maintenance. Crowding distance and clustering operator are two common operators, and they will become computationally expensive when dealing with MaOPs. Visualization of the approximated front is the third difficulty which makes decision-makers more confused in evaluating one algorithm.

In recent years, many researchers have paid attention to the above challenges. Although the third difficulty cannot be avoided effectively, some algorithmic changes to the existing methods may be possible for the first two challenges. According to the characteristics of some recently proposed many-objective optimization algorithm, they can be divided into the following three categories.

1. The first category is to modify the classical Pareto dominance to enhance the selection pressure toward the true front. This idea has been widely used in many papers, and many variants of dominance have been proposed. Laumanns et al. (2002) proposed $\varepsilon$-dominance. Deb et al. (2005) designed a steady-state algorithm ($\varepsilon$-MOEA) which was based on the conception of $\varepsilon$-dominance and achieved a well-distributed and well-converged set of solutions. Yang et al. (2013) presented grid dominance to strengthen the selection pressure toward the optimal direction and then applied it to multi-objective evolutionary algorithm. The above modifications aim to enlarge the dominating area, and thus, more solutions are dominated. Zou et al. (2008) proposed another variant of Pareto dominance called *L*-dominance, which not only takes into account the number of improved objective values, but also considers the values of improved objective functions if all objectives have the same importance. Experimental results demonstrated that the new algorithm based on *L*-dominance obtained good performance.
2. The second type is to reduce the impact of diversity maintenance. Adra and Fleming (Adra and Fleming 2011) proposed a diversity management mechanism, namely DMI, to determine whether or not activate diversity promotion which is deactivated when population is excessively diverse. In terms of both convergence and diversity, evolutionary multi-objective optimization algorithm-based DMI shows excellent performance. Li et al. (2010) employed grid dominance to define an adaptive neighborhood for each individual, whose size varies with the number of objectives. Moreover, adaptive neighborhood and average ranking are integrated to pick out well-converged individuals and prohibit or postpone the archive of adjacent individuals. Similar to (Li et al. 2010), Yang et al. (2013) modified grid dominance and further proposed grid ranking, grid crowding distance and grid coordinate point distance to maintain an extensive and uniform distribution among solutions. Li et al. (2014) developed a general modification of density estimation (SDE) to make Pareto-based algorithms suitable for MaOPs. Different from the traditional density estimation that only concerns the distribution of individuals in the population, SDE shifted the position of other solutions according to convergence comparison with the current solution. Thus, both the distribution and the convergence information of the solutions were considered.
3. The third category is intuitive, that is to say, it employs aggregation functions to differentiate many-objective solutions. One of the representative algorithms is the multi-objective evolutionary algorithm based on decomposition (MOEA/D) (Zhang and Li 2007). The main idea of MOEA/D is to decompose a complex many-objective optimization problem into a number of scalar optimization subproblems using aggregation functions, and each subproblem is then optimized simultaneously. Different from MOEA/D, Hughes (Huband et al. 2006) employed vector angle distance scaling and weighted Tchebycheff methods to rank individuals and then proposed multiple single-objective Pareto sampling (MSOPS) which can perform a parallel search of multiple conventional target vector-based optimization. Based on NSGA-II (Deb et al. 2002b), Deb and Jain suggested a reference point-based many-objective evolutionary algorithm framework (NSGA-III) (Deb and Jain 2014) which emphasized population members that were non-dominated, yet close to a set of supplied reference points. These discussed ideas are effective, and they can avoid dominance relationship which is a great challenge for MaOPs.

There are also a lot of other many-objective optimization algorithms, which employ new ideas different from the above three categories. For example, to tackle the problem of high computational effort required for hyper volume calculation, Bader and Zitzler (2011) used Monte Carlo simulation to approximate the exact hyper volume values. The actual indicator values are not important, but rather the rankings of solutions induced by hyper volume indicator. Zitzler and Kunzli (2004) analyzed how preference information of decision-maker can be integrated into searching process and further proposed to use predefined optimization goal to measure the contribution of each solution.

Cuckoo search (CS) (Yang and Deb 2010b) is an effective swarm intelligence-based algorithm. Since the introduction of CS, it has attracted much attention due to its simplicity and efficiency in dealing with nonlinear optimization problems and real-world engineering applications. In recent years, many researchers have tried to propose new modified CS for MOPs and practical applications (Zhang et al. 2018). However, to the best of our knowledge, no modified CS for MaOPs has been reported in the literature. In this paper, we try to modify CS for MaOPs. First, we remove the global best solution in global search because there is no best solution in MaOPs and then add a new factor to global search. Second, we employ non-dominated sorting and the strategy of reference points in NSGA-III to update the population which is combination of the parent and offspring individuals. To verify the performance of the proposed algorithm, two well-known benchmark sets DTLZ and WFG are utilized in the experiments.

The organization of this paper is as follows. In Sect. 2, the standard CS is briefly introduced. The literature of CS for MOPs is given in Sect. 3. The proposed approach is described in Sect. 4. In Sect. 5, experimental results and analysis are presented. Finally, the work is concluded and summarized in Sect. 6.

## 2 Standard Cuckoo search

Cuckoo search (CS) (Yang and Deb 2010a, b) was firstly proposed by Yang in 2009, which was inspired by breeding behavior of cuckoo. Cuckoos are very interesting birds. They do not engage the obligate brood parasitism directly, but lay their eggs in other species' nests. Although these eggs may be very similar to the eggs of the host birds in appearance, there is still a fraction of some eggs being discovered by host birds. Under this situation, these host birds can either throw these alien eggs away, or abandon the nests. To describe the phenomenon, Yang and Deb idealized the following three rules (Yang and Deb 2010b):

1. Each cuckoo lays one egg at a time and dumps it in a randomly selected nest.
2. The nests with quality egg are carried over to the next generation.
3. Once the alien eggs (solutions) are discovered by host birds, the nests will be abandoned with a probability $p_a \in [0, 1]$.

In fact, the standard CS not only mimics the behavior of cuckoos, but also employs Lévy flight to enhance search ability. Lévy flight is a random walk process. Some studies showed that flight behavior of various birds demonstrates typical characteristic of Lévy flight (Barthelemy et al. 2008; Reynolds and Frye 2007).

In CS, a solution $X_i$ is for cuckoo $i$, and it can be generated with the following Lévy flight:

$$X_i(t') = X_i(t) + \alpha \oplus \text{Levy}(\lambda) \tag{1}$$

where $\alpha > 0$ is the step size, which should be related to the scale of specified problems. The product $\oplus$ is entry-wise multiplications. The step size $\alpha$ can be defined as follows:

$$\alpha = \alpha_0 \times (X_i(t) - X_{\text{best}}) \tag{2}$$

where $\alpha$ = 0.01 and $X_{\text{best}}$ is the optimal position in the population.Levy($\lambda$) satisfies the following Lévy distribution.

$$\text{Levy}(\lambda) \sim u = t^{-\lambda} \tag{3}$$

where $1 < \lambda < 3$. Essentially, Eq. (3) has an infinite variance with an infinite mean. Thus, the steps follow a random walk process with a power-law step-length distribution with a heavy tail.

However, once the alien eggs are discovered by host birds, they may be abandoned with a probability $p_a$. Under this situation, new eggs can be regenerated with the following way:

$$X_i(t+1) = X_i(t') + r \times (X_k(t') - X_j(t')), \tag{4}$$

where $X_k(t')$ and $X_j(t')$ are two randomly selected positions from the whole population. $r$ is a random value ranging from 0 to 1. To have a clear description of the standard CS, we present its pseudo-code in Algorithm 1.

**Algorithm 1:** Standard CS

```
Begin
  Objective function $F(X)$ , $X = (x_1, x_2, ..., x_d)^T$ ;
  Initialize a population of $N$ host nests $X_i (i = 1, 2, ..., N)$ ;
While (stop criteria is not met)
    Get a cuckoo(say $i$) randomly by Lévy flights with Eq. (1);
    Evaluate its fitness/quality $F_i$ ;
    Randomly select a nest among $N$ (say $j$);
If ( $F_i > F_j$ )
    Replace $j$ with the new solution;
End if
    Abandon a fraction ( $p_a$ ) of the worse nests;
    [and build new ones at new locations with Eq.(4)]
    Keep the best solutions (or nests with quality solutions);
    Rank the solutions and find the current best;
End while
  Output the results;
End
```

## 3 Related work

Cuckoo search (CS) (Yang and Deb 2010a, b) algorithm was firstly proposed by Yang in 2009, which was initially designed for single-objective optimization problems. Since then, CS has got a lot of recognitions for its excellent performance in handling complex problems. In fact, many practical engineering problems are MOPs, and some of them have more than three objectives. In the past decades, many efforts have been made to modify CS for MOPs. In this section, a brief review of CS for MOPs is presented.

According to the No-Free-Lunch theorem (Wolpert and Macready 1997), no one algorithm can deal with all kinds of problems effectively. Thus, hybridization of multiple strategies may improve the performance of cuckoo search. Chandrasekaran and Simon (2012) employed fuzzy set theory to create the fuzzy membership search domain and proposed a hybrid cuckoo search algorithm for solving multi-objective unit commitment problem. Rani et al. (2013) combined a modified cuckoo search (MCS) with particle swarm optimization (PSO) and genetic algorithm (GA) in weighted-sum multi-objective optimization. Coelho et al. (2013) presented an improved multi-objective cuckoo search (IMCS), which employs a nearest neighbor density estimation method to obtain the density value for selecting the global best nest. Zhang et al. (2018) proposed a hybrid multi-objective cuckoo search with dynamical local search (HMOCS), in which non-dominated sorting is employed to generate the Pareto front and a dynamical local search is used to enhance the local search.

To solve multi-objective job shop scheduling problem, Hanoun et al. (2012) developed a Pareto archived multi-objective cuckoo search (PAMOCS) to find the set of non-dominated Pareto-optimal solutions. Wang et al. (2012) combined cuckoo search with non-dominated sorting procedure of NSGA-II to solve the optimal design problems of water distribution systems. Zhou et al. (2016, proposed a multi-objective discrete cuckoo search algorithm with local search (MDCL) for community detection problem. In MDCL, the nest location updating strategy and abandon operator of cuckoo are redefined in discrete form, while a local search strategy and a clone operator are proposed to obtain the optimal initial population. To minimize heat transfer and fluid friction irreversibility degrees in plate-fin heat exchangers (PFHEs), Wang and Li (2015) proposed an improved multi-objective cuckoo search algorithm (IMOCS), in which a non-uniform mutation operator is used to create a perfect balance between exploration and exploitation. Moreover, a differential evolution operator is employed to boost cooperation and information exchange among the groups and enhance the quality of convergence.

The above CS variants have shown good performance in dealing with MOPs. With development of society, many MOPs become more and more complex and some of them are MaOPs (Raja et al. 2017; Tozer et al. 2017), which have more than three objectives. For MaOPs, the above-mentioned CS variants are not effective. Moreover, to the best of our knowledge, CS and its modifications have not been applied to MaOPs so far. In this paper, we try to design a hybrid many-objective CS to challenge MaOPs.

## 4 Proposed approach

To solve MaOPs, this paper proposes a hybrid many-objective CS (HMaOCS), which is a combination of modified CS, non-dominated sorting and the strategy of reference points. The standard CS is modified to adapt for MaOPs. Non-dominated sorting is an effective method, which is used to ensure the convergence. The strategy of reference points used in NSGA-III is employed to obtain a uniform distribution of solutions.

In single-objective optimization algorithm, any two solutions can be compared directly and the global best solution can also be easily identified. However, in many-objective optimization algorithms, there exist two relationships: dominated and non-dominated. That is to say, there does not exist the global best solution and the comparison between two solutions is more complex. Therefore, based on the above analysis, two problems should be addressed before modifying CS for MaOPs. The first problem is how to modify the single-objective CS for MaOPs, since there exists the global best solution in Lévy flight, and the second problem is how to update the population for ensuring convergence and diversity in each iteration. The following two parts will address the above two problems, respectively.

### 4.1 Modified CS

In Eq. (2), there exists a global best solution. To incorporate Lévy flight into MaOPs, the new updating equation is redefined as follows:

$$X_i(t') = X_i(t) + 0.01 \times \alpha \times L \times r \times [(X_{\text{upper}} - X_{\text{lower}}) \times r_1 \times r_2] \tag{5}$$

where $\alpha > 0$ is the scale of the movement controlling the moving step and generally equal to 1.0. $L$ is a random number based on Lévy distribution, and $r$ is a random number sampling with standard Gaussian distribution $N(0,1)$. $X_{\text{upper}}$ is the upper bound of variable and $X_{\text{lower}}$ is the lower bound of variable.$r_1$ is a controlling factor, which can be defined as follows:

$$r_1 = 1 - \frac{C_{\text{evaluated}}}{C_{\text{evaluation}}} \tag{6}$$

where $C_{\text{evaluation}}$ is the total number of evaluations and $C_{\text{evaluated}}$ is the current number of evaluations. By employing $(X_{\text{upper}} - X_{\text{lower}}) \times r_1$, individual $X_i(t)$ not only can converge with the evolvement of algorithm, but also can avoid being affected by other individuals. However, in practical problems, each dimension of $(X_{\text{upper}} - X_{\text{lower}}) \times r_1$ is not equally important for improving individual $X_i(t)$. Hence, we assume $X_i(t) = (x_i^1, x_i^2, \ldots, x_i^{D-1}, x_i^D)$ and the corresponding objectives $F(X_i(t)) = (f_1(X_i(t)), f_2(X_i(t)), \ldots, f_M(X_i(t)))$, where $D$ is the number of variables and $M$ is the number of objectives. As shown in Fig. 1, each dimension of $X_i(t)$ may be involved in $f_1(X_i(t))$, and meanwhile only $x_i^{D-2}$,$x_i^{D-1}$ and $x_i^D$ may be involved in $f_{M-1}(X_i(t))$. In this case, we introduce a parameter $r_2 = (r^1, r^2, \ldots, r^i \ldots, r^D)$, where $r^i$ can be randomly set to $-1$, 0, or 1. $r^i$=0 means that the $i$th dimension will not be changed, and $r^i$= 1 or $-1$ means that the $i$th dimension will be updated accordingly.

The probability $p_a$ plays an important role in updating individuals. In the standard CS and its most modifications, $p_a$ is set to 0.25 (Yang and Deb 2010a, b; Zhang et al. 2018). However, for MaOPs, $p_a$ = 0.25 may be not the best choice. Thus, to have a suitable $p_a$ for HMaOCS, we will systematically investigate it in later experiments.

For Eq. (4), we further modify it as follows:

$$X_i(t+1) = X_i(t') + r \times r_3 \times (X_k(t') - X_j(t')) \tag{7}$$

where both $X_k(t')$ and $X_j(t')$ are two individuals randomly selected from the current population, and $r$ ranges from 0 to 1. $r_3 = (r^1, r^2, \ldots, r^i \ldots, r^D)$ is defined as follows:

$$r^i = \begin{cases} 0, 0 < \text{rand} \le \dfrac{1}{M} \\ 1, \dfrac{1}{M} < \text{rand} \le 1 \end{cases} \tag{8}$$

where rand is a random value from 0 to 1 and $M$ is the number of objectives.$r^i = 0$ means $i$th dimension $X_i(t')$ will not be updated, and $r^i = 1$ means it will be changed

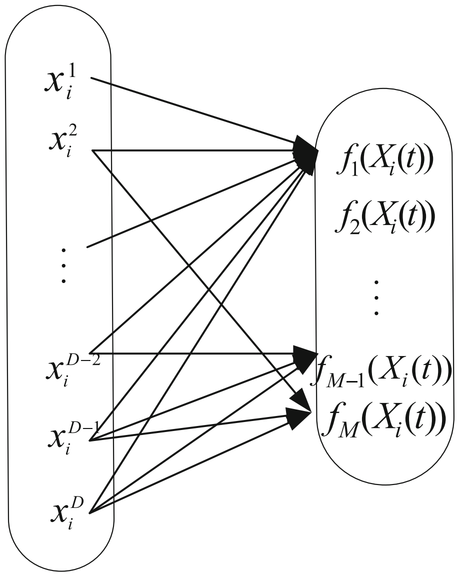


**Fig. 1** Illustration of each dimension

accordingly. Thus, by introducing the parameter $M$, Eq. (8) can not only dynamically adjust the probability, but also handle MaOPs with various numbers of objectives.

## 4.2 Population updating

In many-objective optimization algorithms, there are two important goals which must be considered. One is to make the final solutions as close to the true front as possible, and the other is to get an even distribution of solutions. Convergence and diversity are two main factors for many-objective optimization algorithms. Thus, to ensure good convergence and diversity, we employ non-dominated sorting and the strategy of reference points in NSGA-III (Deb and Jain 2014; Jain and Deb 2014) to update the population.

Assume that $P_t$ and $Q_t$ are the parent and offspring population at the $t$th generation, respectively. Both of them have $N$ individuals. Then, the next step is to select the best $N$ individuals from the combined population $R_t = P_t \cup Q_t$. To implement this task, the procedure of non-dominated sorting is employed. First, the combined population is sorted into multiple non-dominated levels ($F_1, F_2$ and so on). Individuals from non-dominated level 1 to level $l$ are included in the new population $S_t$ until $|S_t| = N$ or $|S_t| > N$ for the first time. If $|S_t| > N$, the level $l$ will be included partially. Assume the last included non-dominated level is level $l$, and the number of the accepted individual in level $l$ is $K = N - |F_1 \cup F_2 \cup \cdots \cup F_{l-1}|$. As for how to select the $K$ individuals from level $l$, we employ the strategy of reference points used in NSGA-III (Deb and Jain 2014; Jain and Deb 2014).

To make reference points [which can be predefined in a structured manner (Das and Dennis 2006)] and objective values have an identical range, they are firstly normalized. Then, the ideal point of the set becomes the zero vector. Each individual in $S_t$ is associated with a reference point according to the proximity of the member with a reference line obtained by joining the ideal point with the reference point. This operation can determine the number of the population members associated with each supplied reference point in $S_t \backslash F_l$. A niching procedure is conducted to select population members from $F_l$ which are not included in $S_t \backslash F_l$. The reference points with least number of association in $S_t \backslash F_l$ are looked for an associated point in $F_l$. Thus, individuals in $F_l$ are then selected one at a time until $|P_{t+1}| = N$. Although the strategy of reference points has a computational complexity of $O(N^2 \log N)$ where $N$ is population size, it helps to deal with MaOPs. The pseudo-code of population updating is presented in Algorithm 2.

**Algorithm 2:** Population Updating

```
Input: supplied reference points Z, combined population $R_t = P_t \cup Q_t$
$S_t = \varnothing, i = 1$
$(F_1, F_2, \ldots) = Non\_dominated\_sort(R_t)$
While $(|S_t| < N)$
$S_t = S_t \cup F_i, i = i + 1$
End while
The last front to be included: $F_l = F_i$
If $|S_t| = N$
$P_{t+1} = S_t$, break
Else
$P_{t+1} = \cup_{j=1}^{l-1} F_i$
    Individuals to be selected from $F_l$: $K = N - |P_{t+1}|$
    Normalize supplied reference points and objective values: $Z^r = Normalize(f^n, S_t, Z)$
    Associate each individual $s$ of $S_t$ with a reference point: $[\pi(s), d(s)] = Associate(S_t, Z^r)$
    % $\pi(s)$ is the closet reference point, $d(s)$ is the distance between $s$ and $\pi(s)$
    Compute the niche count of reference point $\rho_j (j \in Z^r)$
    Choose $K$ individuals one at a time from $F_l$ to fill $P_{t+1}$: $Niching(K, \rho_j, \pi, d, Z^r, F_l, P_{t+1})$
End if
Output: $P_{t+1}$
```

## 4.3 Framework of HMaOCS

Based on modified CS and population updating method, we give the pseudo-code of our proposed HMaOCS in Algorithm 3.

**Algorithm 3:** Proposed Approach (HMaOCS)

```
Begin
Initialize the positions with $N$ host nests and corresponding parameters;
While (stop criteria is not met)
        Update the nest position for each cuckoo with Eq. (5);
        Executethe population updating method;
If $rand() > p_a$
        Re-update the position of corresponding cuckoo with Eq.(7);
End if
Execute the population updating method;
End while
      Output the results;
End
```

From Algorithm 3, it can be seen that there is no guidance information (the global best individual) in Eq. (5), and each individual in the population is not affected by any other individuals. When conducting the population updating method, the new population is generated from the combined parent and offspring populations to ensure better convergence and diversity. A fraction ($p_a$) of the population is updated according to Eq. (7). In fact, Eq. (7) performs a local search to focus on the potential individuals around the current individuals. The newly generated population is then combined together with the former population. Obviously, the population is updated twice. The first update is after the Lévy flight, and the second update is after Eq. (7). The reason for the two updates is that they can prevent the loss of individuals with better convergence, as well as keep an even distribution of population.

## 5 Experimental results and analysis

In order to verify the performance of our approach HMaOCS, two well-known benchmark sets are utilized, including DTLZ (Deb et al. 2002a) and WFG (Hughes 2003) with 2, 3, 4, 6, 8 and 10 objectives. The whole experiments consist of two parts:

1. Investigation of the effects of $p_a$ on the performance of HMaOCS;
2. Comparison of HMaOCS with five other famous many-objective optimization algorithms.

In these experiments, inverted generational distance (IGD) (Zou et al. 2008) is used as a performance indicator. The involved algorithms are listed as follows.

- MOEA/D (Zhang et al. 2018);
- NSGA-III (Deb and Jain 2014);
- KnEA (Zhang and Li 2007);
- GrEA (Jain and Deb 2014);

- HypE (Bader and Zitzler 2011);
- Proposed HMaOCS.

## 5.1 Parameter settings

To have a fair comparison, the same parameter settings for all compared algorithms are used according to their original papers. Authors can also refer to paper (Zhang et al. 2015) which has listed most of the parameters used in this paper.

1. Population size ($N$) and parameters ($p_1$, $p_2$) in the algorithms: Table 1 presents the parameters $p_1$, $p_2$, and $N$ under different objectives. The parameters $p_1$ and $p_2$ are used to control the numbers of references points in NSGA-III and MOEA/D (Deb and Jain 2014; Zhang and Li 2007). Note that the strategy of two-layered references points is used as developers suggested in original papers to generate uniformly distributed weighted vectors (Deb and Jain 2014).
2. Test instances: To validate the performance of HMaOCS, two benchmark sets DTLZ and WFG are employed. Table 2 shows some parameter settings on the DTLZ benchmark set with seven test instances DTLZ1 to DTLZ7, where $M$ is the number of objectives, $D$ is the number of variables, and *MaxIter* is the maximum number of iterations. Table 3 presents some parameter settings on the WFG benchmark set with nine test instances WFG1 to WFG9.
3. Stopping condition and number of runs: For each test instance, each algorithm is run 20 independent times on the same machine with Intel Core i5-2400 3.10 GHz CPU, 6.00 GB memory, and Windows 7 operating system with Matlab 7.9. In this paper, the maximum number of iterations (*MaxIter*) is used as the stopping criterion. For the DTLZ test suit, the *MaxIter* is listed in Table 2. For WFG1 and WFG2, the *MaxIter* is set to 1000 and 700, respectively. For WFG3 to WFG9, the *MaxIter* is set to 250.
4. Performance assessment: Inversed Generational Distance (IGD) (Li and Zheng 2009; Zou et al. 2008) is a widely used indicator for evaluating many-objective optimization algorithms. IGD can not only evaluate the convergence (closeness to the true Pareto-optimal fronts), but also the distribution of obtained final solutions. Let $P^*$ be a set of uniformly distributed points in the objective space along the true Pareto front (PF). Let $P$ be a set of obtained solutions. Then, the IGD can be defined by

$$IGD(P, P^*) = \frac{\sum_{v \in P^*} dist(v, P)}{|P^*|}, \quad (9)$$

where $dist(v,P)$ is the minimum Euclidean distances between $v$ and point in $P$. Note that $P^*$ is a set of sampling points. In general, the number of $P^*$is predefined. For all test problems, the number (the closest integer to 500 among the possible values) of reference points is used for the IGD calculation.
5. Compared algorithms: Table 4 shows parameter settings for NSGA-III and MOEA/D (Deb and Jain 2014; Zhang et al. 2018), where $D$ is the number of decision variables. The parameter *div* in Table 5 is the number of divisions in each dimension in GrEA. To get the best

**Table 1** Parameter settings of population size

| Number of objectives ($M$) | Parameters ($p_1$, $p_2$) | Population size ($N$) |
|---|---|---|
| 2 | (99, 0) | 100 |
| 3 | (12, 0) | 91 |
| 4 | (7, 0) | 120 |
| 6 | (4, 1) | 132 |
| 8 | (3, 2) | 156 |
| 10 | (3, 2) | 275 |

**Table 2** Parameter settings on DTLZ1 to DTLZ7

| Test instances | $M$ | $k$ | $D$ | *MaxIter* |
|---|---|---|---|---|
| DTLZ1 | 2, 3, 4, 6, 8, 10 | 5 | $M-1+k$ | 700 |
| DTLZ2 | 2, 3, 4, 6, 8, 10 | 10 | $M-1+k$ | 250 |
| DTLZ3 | 2, 3, 4, 6, 8, 10 | 10 | $M-1+k$ | 1000 |
| DTLZ4 | 2, 3, 4, 6, 8, 10 | 10 | $M-1+k$ | 250 |
| DTLZ5 | 2, 3, 4, 6, 8, 10 | 10 | $M-1+k$ | 250 |
| DTLZ6 | 2, 3, 4, 6, 8, 10 | 10 | $M-1+k$ | 250 |
| DTLZ7 | 2, 3, 4, 6, 8, 10 | 20 | $M-1+k$ | 250 |

**Table 3** Parameter settings on WFG1 to WFG9

| $M$ | $K$ | $L$ | $D$ |
|---|---|---|---|
| 2 | 4 | 10 | $K+L$ |
| 3 | 4 | 10 | $K+L$ |
| 4 | 6 | 10 | $K+L$ |
| 6 | 5 | 10 | $K+L$ |
| 8 | 7 | 10 | $K+L$ |
| 10 | 9 | 10 | $K+L$ |

**Table 4** Parameter settings in NSGA-III and MOEA/D

| Parameters | NSGA-III | MOEA/D |
|---|---|---|
| SBX probability, $p_c$ | 1 | 1 |
| Polynomial mutation probability, $p_m$ | $1/D$ | $1/D$ |
| $\eta_c$ | 30 | 20 |
| $\eta_m$ | 20 | 20 |

**Table 5** Parameter settings of *div* in GrEA on DTLZ and WFG test suits

| Test instance | $M = 2$ | $M = 3$ | $M = 4$ | $M = 6$ | $M = 8$ | $M = 10$ |
|---|---|---|---|---|---|---|
| DTLZ1 | 55 | 10 | 10 | 10 | 10 | 11 |
| DTLZ2 | 45 | 16 | 10 | 10 | 8 | 12 |
| DTLZ3 | 45 | 11 | 11 | 11 | 10 | 11 |
| DTLZ4 | 55 | 10 | 10 | 8 | 8 | 12 |
| DTLZ5 | 55 | 35 | 35 | 14 | 11 | 11 |
| DTLZ6 | 55 | 36 | 36 | 20 | 20 | 20 |
| DTLZ7 | 16 | 8 | 9 | 6 | 6 | 4 |
| WFG1 | 45 | 8 | 8 | 9 | 7 | 10 |
| WFG2 | 55 | 11 | 11 | 11 | 11 | 11 |
| WFG3 | 55 | 16 | 18 | 18 | 16 | 22 |
| WFG4-9 | 45 | 10 | 10 | 9 | 8 | 12 |

**Table 6** Parameter settings of *T* in KnEA on DTLZ and WFG test suits

| Test instance | $M = 2$ | $M = 3$ | $M = 4$ | $M = 6$ | $M = 8$ | $M = 10$ |
|---|---|---|---|---|---|---|
| DTLZ1 | 0.6 | 0.6 | 0.6 | 0.2 | 0.1 | 0.1 |
| DTLZ2 | 0.6 | 0.6 | 0.5 | 0.5 | 0.5 | 0.5 |
| DTLZ3 | 0.6 | 0.5 | 0.5 | 0.2 | 0.1 | 0.1 |
| DTLZ4 | 0.6 | 0.6 | 0.5 | 0.5 | 0.5 | 0.5 |
| DTLZ5 | 0.6 | 0.5 | 0.5 | 0.5 | 0.3 | 0.3 |
| DTLZ6 | 0.6 | 0.5 | 0.5 | 0.4 | 0.3 | 0.3 |
| DTLZ7 | 0.6 | 0.6 | 0.5 | 0.5 | 0.5 | 0.4 |
| WFG4 | 0.6 | 0.5 | 0.5 | 0.5 | 0.3 | 0.3 |
| WFG9 | 0.6 | 0.6 | 0.5 | 0.5 | 0.3 | 0.3 |
| Others | 0.6 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |

setting of *div,* we tested many values by the suggestion of (Zitzler et al. 2001), and the values that gain the best performance are listed in Table 5. Table 6 shows the settings of *T* in KnEA, which is used to control the ratio of knee points to the non-dominated solutions.

## 5.2 Investigation of the effect of $p_a$ on the performance of HMaOCS

As discussed in Sect. 5.2, the parameter $p_a$ plays an important role in selecting the fraction of the current population. In single-objective optimization algorithms, many references (Yang and Deb 2010a, b; Zhang et al. 2018) have suggested some proper settings of $p_a$. However, few papers have been reported in many-objective optimization. Hence, to investigate the effect of the parameter $p_a$, HMaOCS is tested on the DTLZ test suit with different $p_a$ values. In the experiments, $p_a$ is set to 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8 and 0.9, respectively. $p_a = 0$ means that all individuals are selected. When $p_a = 1$, all individuals will not be updated according to Eq. (7). Therefore, $p_a = 1$ will not be investigated.

Table 7 presents the mean IGD results of HMaOCS with different $p_a$ on DTLZ1-DTLZ7, where the best results are shown in boldface. From the results, we can roughly draw the conclusion that $p_a = 0.6$, 0.7 and 0.8 have similar performance, and they obtain better IGD values than other $p_a$ settings.

To have a clear comparison, we use Friedman test to calculate the mean ranks of HMaOCS with different $p_a$ values. Friedman test is commonly used in intelligent optimization field for comparing different algorithms and showing their performance. The resulted values, namely the ranks, indicate the compared results. The smallest rank corresponds to the best algorithm. Detailed basic explanations for Friedman test can be found in paper (Sun et al. 2014). The Friedman test procedure can be explained as follows. Firstly, the average IGD results of HMaOCS with different $p_a$ settings on test problems with various numbers of objectives are computed (As presented in Table 7). Then, the rank values of different $p_a$ settings are computed using the function Friedman test in IBM SPSS Statistics. (IBM SPSS Statistics is a widely used statistical software.) The resulted rank values are then presented and listed in Table 8. Table 8 tabulates the corresponding rank results of HMaOCS with different $p_a$, where the best rank is shown in boldface. As shown in Table 8, $p_a = 0.7$ obtains the smallest rank, which demonstrates that $p_a = 0.7$ is the relatively best setting for the test suite.

## 5.3 Comparison of HMaOCS with other algorithms on the DTLZ test suit

As mentioned before, the second experiment focuses on the comparison of HMaOCS with five other well-known many-objective optimization algorithms. Table 9 presents the mean IGD results of MOEA/D, NSGA-III, KnEA, GrEA, HypE and HMaOCS on the DTLZ test suite, where the best results are shown in boldface. The comparison results among HMaOCS and other algorithms are summarized as "*w/l/t*", which means that HMaOCS wins in *w* functions, loses in *l* functions, and ties in *t* functions.

From the last line of Table 9, it is obvious that HMaOCS outperforms KnEA, GrEA and HypE. However, HMaOCS is slightly worse than MOEA/D, which is famous for its high computational efficiency. The same results also appear between HMaOCS and NSGA-III. In terms of each instance, HMaOCS is superior over MOEA/D on DTLZ2 and DTLZ4 with 2 and 4 objectives. The possible reason is that DTLZ4 is a series of non-uniform

**Table 7** IGD results of HMaOCS with different $p_a$ on DTLZ1 to DTLZ7

| Test instance | $M$ | Parameter $p_a$ | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
| DTLZ1 | 2 | 3.69E−02 | **1.79E−03** | 3.69E−02 | 1.78E−01 | 2.84E−01 | 1.07E−01 | 1.07E−01 | 1.78E−01 | 2.48E−01 | 1.03E+00 |
| | 4 | 4.25E−02 | 4.20E−02 | 4.17E−02 | 6.37E−02 | 4.13E−02 | 4.13E−02 | **4.12E−02** | 8.57E−02 | 1.08E−01 | 1.09E−01 |
| | 6 | 9.35E−02 | 8.90E−02 | 8.63E−02 | 8.43E−02 | 8.29E−02 | 8.22E−02 | 8.21E−02 | **8.21E−02** | 8.22E−02 | 1.47E−01 |
| | 8 | 1.88E−01 | 1.37E−01 | 1.14E−01 | 1.07E−01 | 1.04E−01 | 1.01E−01 | 9.96E−02 | **9.88E−02** | 1.15E−01 | 9.95E−02 |
| | 10 | 2.36E−01 | 1.58E−01 | 1.37E−01 | 1.27E−01 | 1.17E−01 | 1.15E−01 | 1.13E−01 | 1.12E−01 | **1.11E−01** | 1.11E−01 |
| DTLZ2 | 2 | 4.36E−03 | 4.27E−03 | 4.14E−03 | 4.09E−03 | 4.08E−03 | 4.05E−03 | 4.06E−03 | **4.02E−03** | 4.03E−03 | 4.17E−03 |
| | 4 | 2.06E−01 | 1.81E−01 | 1.68E−01 | 1.53E−01 | 1.41E−01 | 1.33E−01 | 1.28E−01 | **1.24E−01** | 1.24E−01 | 1.25E−01 |
| | 6 | 6.21E−01 | 5.73E−01 | 5.29E−01 | 4.71E−01 | 4.10E−01 | 3.59E−01 | 3.13E−01 | 2.88E−01 | **2.75E−01** | 2.75E−01 |
| | 8 | 8.09E−01 | 7.66E−01 | 6.81E−01 | 6.84E−01 | 5.80E−01 | 5.63E−01 | 4.68E−01 | 4.16E−01 | **3.87E−01** | 3.94E−01 |
| | 10 | 8.76E−01 | 9.06E−01 | 7.97E−01 | 7.51E−01 | 6.92E−01 | 6.66E−01 | 6.13E−01 | 5.15E−01 | 5.18E−01 | **4.85E−01** |
| DTLZ3 | 2 | 2.30E+00 | 1.20E+00 | 8.02E−01 | **1.10E+00** | 1.30E+00 | 1.70E+00 | 2.40E+00 | 5.50E+00 | 4.10E+00 | 8.20E+00 |
| | 4 | 8.39E−01 | 4.87E−01 | 3.52E−01 | 4.26E−01 | **3.16E−01** | 4.96E−01 | 8.77E−01 | 3.98E−01 | 1.06E+00 | 1.95E+00 |
| | 6 | 1.09E+01 | 6.94E+00 | 4.50E+00 | 2.72E+00 | 7.73E−01 | **2.97E−01** | 3.53E−01 | 3.52E−01 | 4.28E−01 | 5.36E−01 |
| | 8 | 1.94E+01 | 1.53E+01 | 1.08E+01 | 7.24E+00 | 4.36E+00 | 1.98E+00 | 8.43E−01 | **3.77E−01** | 7.42E−01 | 4.21E−01 |
| | 10 | 3.63E+01 | 2.51E+01 | 1.43E+01 | 1.18E+01 | 7.66E+00 | 3.46E+00 | 1.32E+00 | 6.14E−01 | **4.50E−01** | 5.23E−01 |
| DTLZ4 | 2 | 4.04E−03 | 4.04E−03 | 4.03E−03 | 4.01E−03 | 3.99E−03 | 3.98E−03 | **3.98E−03** | 3.98E−03 | 7.78E−02 | 4.02E−03 |
| | 4 | 1.54E−01 | 1.50E−01 | 1.43E−01 | 1.36E−01 | 1.31E−01 | 1.29E−01 | 1.25E−01 | **1.23E−01** | 1.23E−01 | 1.24E−01 |
| | 6 | 5.47E−01 | 5.29E−01 | 4.82E−01 | 4.26E−01 | 3.96E−01 | 3.53E−01 | 3.18E−01 | 2.89E−01 | **2.75E−01** | 2.76E−01 |
| | 8 | 6.74E−01 | 6.61E−01 | 6.13E−01 | 6.08E−01 | 5.40E−01 | 5.04E−01 | 4.58E−01 | 4.00E−01 | 3.72E−01 | **3.69E−01** |
| | 10 | 7.42E−01 | 7.16E−01 | 7.28E−01 | 6.97E−01 | 6.71E−01 | 6.06E−01 | **4.55E−01** | 5.15E−01 | 4.83E−01 | 4.75E−01 |
| DTLZ5 | 2 | 4.34E−03 | 4.26E−03 | 4.18E−03 | 4.10E−03 | 4.09E−03 | 4.06E−03 | 4.04E−03 | **4.03E−03** | 4.03E−03 | 4.14E−03 |
| | 4 | 2.99E−01 | 3.15E−01 | 1.82E−01 | 1.96E−01 | 1.22E−01 | 1.39E−01 | 1.28E−01 | 1.13E−01 | 1.15E−01 | **1.03E−01** |
| | 6 | 6.82E−01 | 7.27E−01 | 6.82E−01 | 7.29E−01 | 7.00E−01 | 6.50E−01 | 6.06E−01 | 5.45E−01 | **4.89E−01** | 5.10E−01 |
| | 8 | 9.83E−01 | 8.32E−01 | 8.98E−01 | 8.56E−01 | 7.74E−01 | 7.66E−01 | 7.52E−01 | 6.95E−01 | 7.04E−01 | **6.12E−01** |
| | 10 | 9.77E−01 | 1.01E+00 | 1.00E+00 | 9.27E−01 | 9.19E−01 | 9.08E−01 | 8.64E−01 | 7.56E−01 | 7.04E−01 | **6.52E−01** |
| DTLZ6 | 2 | **3.97E−03** | 3.97E−03 | 3.97E−03 | 3.97E−03 | 3.97E−03 | 3.97E−03 | 3.98E−03 | 3.99E−03 | 4.01E−03 | 4.32E−03 |
| | 4 | 6.96E−02 | 7.51E−02 | 7.85E−02 | 9.67E−02 | 9.52E−02 | 8.45E−02 | **1.05E−01** | 1.38E−01 | 1.28E−01 | 1.48E−01 |
| | 6 | 1.93E−01 | 2.17E−01 | 2.60E−01 | **1.97E−01** | 2.78E−01 | 2.61E−01 | 2.77E−01 | 2.87E−01 | 2.47E−01 | 6.14E−01 |
| | 8 | 7.79E−01 | **4.70E−01** | 5.14E−01 | 1.08E+00 | 5.93E−01 | 5.66E−01 | 6.45E−01 | 7.99E−01 | 1.12E+00 | 1.12E+00 |
| | 10 | 2.45E−01 | **3.52E−01** | 8.08E−01 | 9.02E−01 | 1.10E+00 | 6.28E−01 | 1.05E+00 | 1.50E+00 | 1.34E+00 | 2.03E+00 |
| DTLZ7 | 2 | 1.16E−02 | 9.96E−03 | 9.47E−03 | 8.68E−03 | 8.26E−03 | 8.29E−03 | **8.05E−03** | 8.24E−03 | 8.72E−03 | 1.09E−02 |
| | 4 | 3.98E−01 | 2.94E−01 | 2.43E−01 | 2.37E−01 | 2.19E−01 | 2.10E−01 | 2.03E−01 | **2.03E−01** | 2.07E−01 | 2.30E−01 |
| | 6 | 1.70E+00 | 1.32E+00 | 8.88E−01 | 8.44E−01 | 7.82E−01 | 7.72E−01 | 7.22E−01 | 7.34E−01 | **7.10E−01** | 7.56E−01 |
| | 8 | 5.09E+00 | 4.50E+00 | 3.27E+00 | 2.94E+00 | 2.40E+00 | 2.18E+00 | 1.77E+00 | **1.42E+00** | 1.47E+00 | 1.84E+00 |
| | 10 | 5.51E+00 | 4.74E+00 | 4.98E+00 | 5.01E+00 | 3.92E+00 | 4.09E+00 | 3.32E+00 | 2.86E+00 | 2.60E+00 | **2.52E+00** |

many-objective optimization problems, and evenly distributed weights used in MOEA/D are unable to effectively deal with them. Both NSGA-III and HMaOCS employ non-dominated sorting and the strategy of reference points, and their differences are the selection and crossover operations. Comparison results illustrate that the selection and crossover operations play a similar role as the modified Lévy flight and Eq. (7). On DTLZ2, DTLZ4 and DTLZ6 with 2, 4 and 6 objectives, HMaOCS performs better than KnEA. Compared with GrEA, HMaOCS works better on DTLZ1, DTLZ3 and DTLZ6 with 4, 6, 8 and 10 objectives. It is obvious that HMaOCS outperforms HypEon DTLZ2, DTLZ4 and DTLZ7. On DTLZ1 with 4, 6, 8 and 10 objectives, HMaOCS also shows better performance than HypE. The above analysis of comparison results demonstrates that HMaOCS is promising in dealing with most many-objective optimization problems.

To have a clear comparison, Fig. 2 presents the obtained solutions of six algorithms on DTLZ4 with 3 objectives. As exhibited in Fig. 2, HMaOCS, MOEA/D and NSGA-III

**Table 8** Rank achieved by the Friedman test

| $p_a$ | 0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | **0.7** | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|---|
| Mean rank | 8.04 | 7.26 | 6.53 | 6.46 | 5.46 | 4.49 | 3.97 | **3.71** | 4.06 | 5.03 |

**Table 9** IGD results of the six algorithms on DTLZ1 to DTLZ7

| Problem | Obj. | HMaOCS | MOEA/D | NSGA-III | KnEA | GrEA | HypE |
|---|---|---|---|---|---|---|---|
| DTLZ1 | 2 | 1.7769e−1 (1.85e−1) | 1.7970e−3 (1.07e−5) | **1.7898e−3 (8.59e−6)** | 3.8334e−3 (1.47e−3) | 1.3013e−1 (1.26e−1) | 1.8563e−3 (3.47e−5) |
| | 4 | 6.3437e−2 (7.01e−2) | **4.1244e−2 (5.62e−5)** | 4.1271e−2 (7.76e−5) | 1.3379e−1 (5.36e−2) | 1.1206e−1 (8.17e−2) | 1.8322e−1 (5.73e−2) |
| | 6 | **8.1825e−2 (2.53e−4)** | 8.1930e−2 (2.43e−4) | 8.1375e−2 (2.55e−4) | 2.9840e−1 (4.38e−2) | 2.2882e−1 (1.18e−1) | 2.9788e−1 (3.99e−2) |
| | 8 | **9.8813e−2 (3.46e−4)** | 9.9135e−2 (2.86e−4) | 1.0721e−1 (1.47e−2) | 3.2239e−1 (1.28e−1) | 2.9043e−1 (7.36e−2) | 3.2690e−1 (3.06e−2) |
| | 10 | **1.1170e−1 (1.22e−3)** | 1.1890e−1 (7.96e−5) | 1.1751e−1 (1.45e−2) | 2.1355e+0 (8.76e−1) | 3.6630e−1 (3.02e−2) | 3.4449e−1 (5.03e−2) |
| DTLZ2 | 2 | **4.0434e−3 (6.03e−5)** | 4.1718e−3 (1.66e−5) | 4.1700e−3 (5.15e−6) | 6.9615e−2 (1.55e−2) | 1.0540e−2 (7.69e−5) | 5.6788e−3 (2.98e−4) |
| | 4 | **1.2451e−1 (4.33e−4)** | 1.2523e−1 (8.78e−6) | 1.3125e−1 (2.79e−5) | 1.4069e−1 (5.28e−3) | 1.3059e−1 (1.32e−3) | 2.6328e−1 (1.31e−2) |
| | 6 | 2.8912e−1 (8.03e−3) | 2.5603e−1 (1.58e−4) | 2.5813e−1 (9.41e−4) | 2.8951e−1 (5.13e−3) | 2.5946e−1 (1.49e−3) | 4.2913e−1 (1.91e−2) |
| | 8 | 4.5212e−1 (8.75e−2) | **3.1578e−1 (2.47e−4)** | 3.2103e−1 (1.02e−3) | 3.8575e−1 (7.41e−3) | 3.5005e−1 (1.23e−3) | 5.9295e−1 (4.61e−2) |
| | 10 | 5.3850e−1 (2.01e−2) | 4.2090e−1 (6.67e−4) | 5.4485e−1 (1.15e−1) | 4.3390e−1 (2.14e−3) | **4.0657e−1 (1.36e−3)** | 7.2874e−1 (5.68e−2) |
| DTLZ3 | 2 | 3.2994e+0 (3.37e+0) | **4.3398e−3 (4.76e−4)** | 4.4918e−3 (7.82e−4) | 1.6118e−1 (2.34e−2) | 1.0309e−2 (4.96e−4) | 5.8365e−3 (4.29e−4) |
| | 4 | 5.9884e−1 (4.58e−1) | **1.2145e−1 (4.97e−4)** | 1.2164e−1 (4.50e−4) | 3.1687e−1 (1.82e−1) | 6.9809e−1 (3.43e−1) | 5.0194e−1 (9.91e−2) |
| | 6 | **2.7625e−1 (2.52e−2)** | 2.8653e−1 (5.82e−4) | 2.7912e−1 (1.75e−3) | 7.1353e−1 (1.45e−1) | 9.5058e−1 (1.76e−1) | 7.2584e−1 (6.27e−2) |
| | 8 | 4.4354e−1 (2.62e−1) | **3.1649e−1 (9.11e−4)** | 1.1111e+0 (1.62e+0) | 4.9112e+1 (2.43e+1) | 1.0987e+0 (1.30e−1) | 9.4646e−1 (6.94e−2) |
| | 10 | 6.0150e−1 (3.06e−2) | **4.2238e−1 (3.10e−3)** | 4.2815e−1 (3.43e−3) | 2.8503e+2 (8.26e+1) | 2.3113e+0 (4.74e−1) | 1.0239e+0 (6.97e−2) |
| DTLZ4 | 2 | **3.9772e−3 (5.08e−6)** | 2.2541e−1 (3.57e−1) | 7.7781e−2 (2.33e−1) | 8.6208e−2 (2.30e−1) | 2.2991e−1 (3.53e−1) | 1.3036e−1 (2.30e−1) |
| | 4 | **1.2348e−1 (7.00e−4)** | 5.0119e−1 (3.20e−1) | 1.8790e−1 (1.40e−1) | 1.3807e−1 (2.56e−3) | 2.6401e−1 (1.67e−1) | 4.6027e−1 (2.24e−1) |
| | 6 | 2.8969e−1 (4.61e−3) | 6.6120e−1 (2.05e−1) | 2.961e−1 (7.28e−2) | 2.8974e−1 (5.57e−3) | **2.6000e−1 (1.47e−3)** | 5.6025e−1 (1.35e−1) |
| | 8 | 4.0701e−1 (9.57e−3) | 6.4731e−1 (8.47e−2) | 3.6434e−1 (8.49e−2) | 3.7364e−1 (2.22e−3) | **3.5162e−1 (2.68e−3)** | 6.3753e−1 (4.09e−2) |
| | 10 | 5.1337e−1 (7.13e−3) | 6.2212e−1 (2.48e−3) | 4.3374e−1 (3.73e−4) | 4.3369e−1 (3.23e−3) | **4.0836e−1 (8.91e−4)** | 7.0431e−1 (2.78e−2) |
| DTLZ5 | 2 | 4.0233e−3 (3.33e−5) | **3.9771e−3 (2.18e−5)** | 4.0282e−3 (7.27e−7) | 6.4147e−2 (1.75e−2) | 1.0487e−2 (2.37e−4) | 5.5337e−3 (2.10e−4) |
| | 4 | 9.0828e−2 (2.18e−2) | **2.7608e−2 (2.89e−4)** | 5.3010e−2 (1.58e−2) | 1.6426e−1 (6.59e−2) | 8.6268e−2 (1.86e−2) | 2.9551e−2 (2.71e−3) |
| | 6 | 5.8432e−1 (7.13e−2) | **2.0996e−2 (3.06e−5)** | 2.6719e−1 (7.88e−2) | 2.6329e−1 (4.74e−2) | 1.9453e−1 (3.47e−2) | 3.7427e−2 (4.27e−3) |
| | 8 | 7.2090e−1 (6.99e−2) | **2.4825e−2 (6.84e−4)** | 3.2512e−1 (1.10e−1) | 3.0040e−1 (5.24e−2) | 2.8614e−1 (5.27e−2) | 4.8291e−2 (7.25e−3) |

Table 9 continued

| Problem | Obj. | HMaOCS | MOEA/D | NSGA-III | KnEA | GrEA | HypE |
|---|---|---|---|---|---|---|---|
| | 10 | 9.1912e−1 (2.63e−1) | **1.8784e−2 (4.66e−7)** | 8.6915e−1 (2.04e−1) | 4.0559e−1 (1.48e−1) | 3.4053e−1 (8.68e−2) | 5.2957e−2 (3.38e−5) |
| DTLZ6 | 2 | **3.9897e−3 (1.53e−5)** | 3.9962e−3 (1.25e−7) | 3.9966e−3 (1.97e−7) | 8.8903e−2 (1.66e−2) | 1.0685e−2 (8.40e−5) | 5.9566e−3 (3.82e−4) |
| | 4 | 1.0822e−1 (4.62e−2) | **2.7266e−2 (2.89e−4)** | 7.4748e−2 (2.06e−2) | 2.0163e−1 (7.31e−2) | 1.8335e−1 (5.62e−2) | 2.0046e−1 (3.15e−2) |
| | 6 | 2.4310e−1 (8.83e−2) | **1.8931e−2 (1.36e−3)** | 6.4673e−1 (3.58e−1) | 5.9790e−1 (3.20e−1) | 2.9522e−1 (4.67e−2) | 1.7257e−1 (5.26e−2) |
| | 8 | 8.6442e−1 (7.40e−1) | **2.5117e−2 (9.92e−4)** | 2.2485e+0 (1.28e+0) | 1.4258e+0 (4.70e−1) | 6.8766e−1 (2.31e−1) | 1.7202e−1 (6.35e−2) |
| | 10 | 1.2304e+0 (1.10e+0) | **1.8686e−2 (1.18e−5)** | 2.6116e+0 (1.64e+0) | 1.5325e+0 (2.65e−1) | 1.2884e+0 (1.29e−1) | 1.1735e−1 (1.37e−2) |
| DTLZ7 | 2 | 8.0007e−3 (4.49e−4) | 1.4379e−1 (2.09e−1) | **6.9478e−3 (1.51e−4)** | 3.4778e−2 (1.21e−2) | 2.8488e−2 (5.35e−3) | 2.6755e−1 (2.27e−1) |
| | 4 | 2.0174e−1 (7.58e−3) | 3.7499e−1 (1.63e−2) | 2.1859e−1 (6.92e−3) | 1.8639e−1 (8.72e−2) | **1.8325e−1 (7.47e−2)** | 1.1795e+0 (3.33e−2) |
| | 6 | 7.1015e−1 (4.78e−2) | 1.1889e+0 (1.15e−1) | 6.1682e−1 (2.88e−2) | 3.9537e−1 (3.72e−2) | **4.3411e−1 (2.82e−2)** | 2.4490e+0 (1.88e−1) |
| | 8 | 1.6054e+0 (4.24e−1) | 1.5648e+0 (2.51e−1) | 8.9065e−1 (9.63e−2) | 7.9986e−1 (4.09e−1) | **7.9983e−1 (5.60e−2)** | 4.0327e+0 (1.13e−1) |
| | 10 | 2.9988e+0 (5.07e−1) | 2.4437e+0 (5.83e−1) | 1.2999e+0 (4.41e−2) | **8.5226e−1 (1.09e−2)** | 2.6085e+0 (6.47e−2) | 5.2127e+0 (1.40e−1) |
| ***w/l/t*** | | | **15/20/0** | **16/19/0** | **21/14/0** | **19/16/0** | **25/10/0** |

perform similar performance regarding the diversity, whereas GrEA, KnEA and HypE fail to keep an even distribution of obtained solutions. The reason may be that the knee points in KnEA are used to enhance the convergence. HypE is also poor because the HV (Deb and Kalyanmoy 2001) used in HypE has a bias toward typical knee points (Jain and Deb 2014). It is easy to understand that HMaOCS performs similar performance as NSGA-III does because both of them employ the same strategy of reference points.

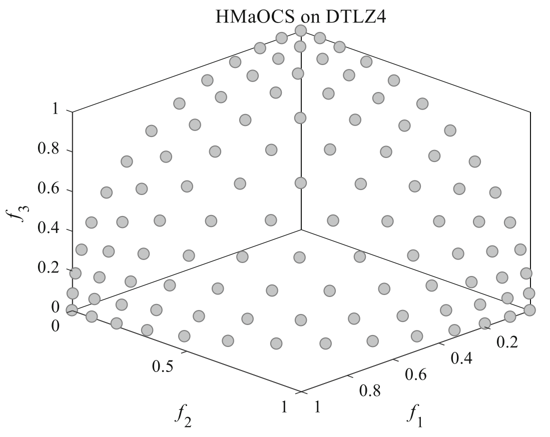
## 5.4 Comparison of HMaOCS with other algorithms on the WFG test suit

Table 10 presents the mean IGD results of MOEA/D, NSGA-III, KnEA, GrEA, HypE and HMaOCS on the WFG test suite, where the best results are shown in boldface. From the last line of Table 10, HMaOCS outperforms MOEA/D on 39 functions, while MOEA/D is better than HMaOCS on 6 functions. We can easily get similar conclusions from comparison results of HMaOCS, KnEA and HypE.

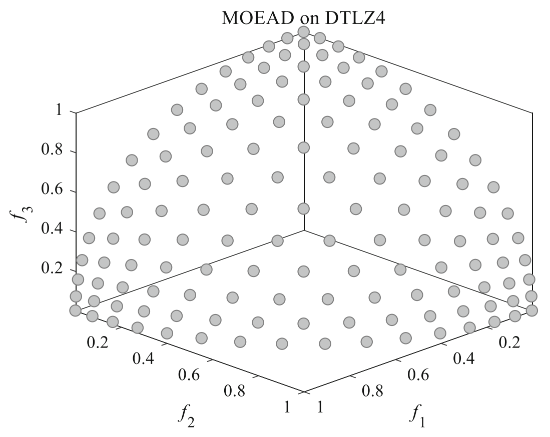
Although HMaOCS is based on the careful modifications of CS and is further incorporated with the strategy of reference points used in NSGA-III, HMaOCS does not show obvious advantage over NSGA-III and wins in only 23 functions. On WFG4 to WFG8, HMaOCS obviously outperforms KnEA and obtains better results on WFG2 with 2, 4, 6 and 8 objectives. Compared to GrEA, HMaOCS performs better on all WFG test instances except for WFG1, WFG8 and WFG9. HypE is better than HMaOCS on WFG3 and other instances with 2 objectives. The above analysis demonstrates that HMaOCS shows excellent performance.

Figure 3 presents the obtained solutions of six algorithms on WFG6 with 3 objectives. In terms of diversity, NSGA-III shows the best performance. HMaOCS has no obvious advantage over MOEA/D, but both of them are slightly worse than NSGA-III. KnEA, GrEA and HypE show the worst performance regarding the distribution of obtained solutions.

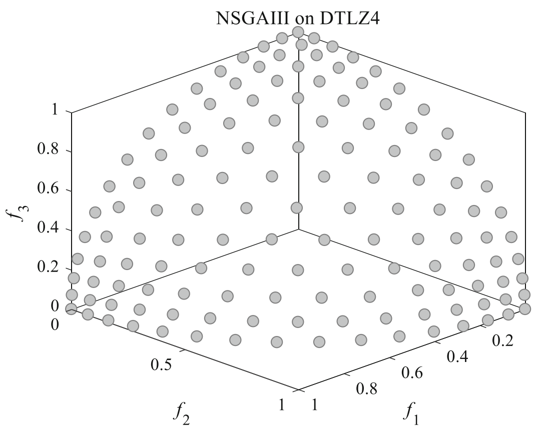
Table 11 summarizes the comparison results between HMaOCS and five other many-objective optimization algorithms on the DTLZ and WFG benchmark sets, where "*w/l/t*" means that HMaOCS wins in *w* functions, loses in *l* functions and ties in *t* functions. From the results, though HMaOCS is slightly worse than MOEA/D on the DTLZ benchmark set, it performs much better than MOEA/D on the WFG benchmark set. The total results show that HMaOCS outperforms MOEA/D. However, both NSGA-III and HMaOCS obtain similar performance. NSGA-III is slightly better than HMaOCS on the DTLZ benchmark set, while HMaOCS outperforms NSGA-III on the WFG



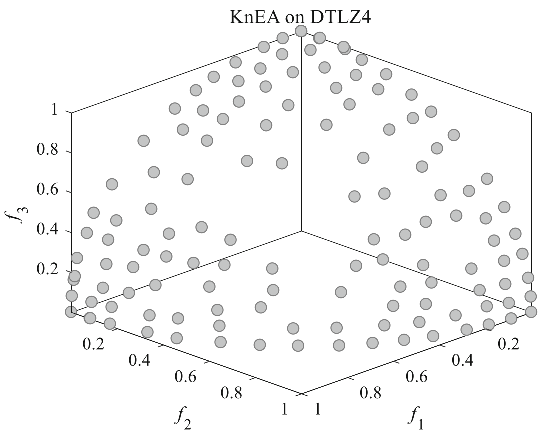
**(a)** Solutions obtained by HMaOCS on DTLZ4



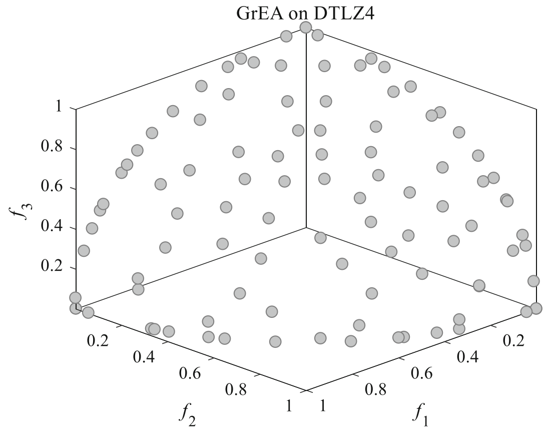
**(b)** Solutions obtained by MOEA/D on DTLZ4



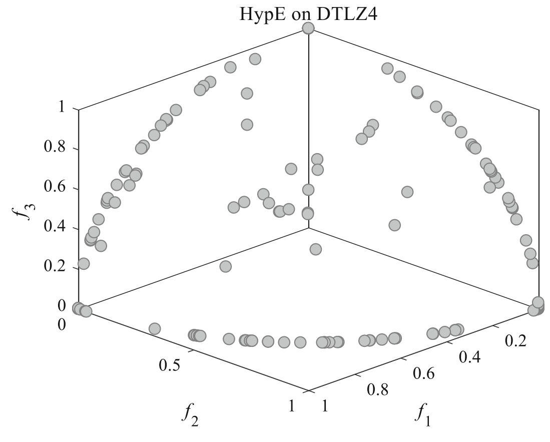
**(c)** Solutions obtained by NSGA-III on DTLZ4



**(d)** Solutions obtained by KnEA on DTLZ4



**(e)** Solutions obtained by GrEA on DTLZ4



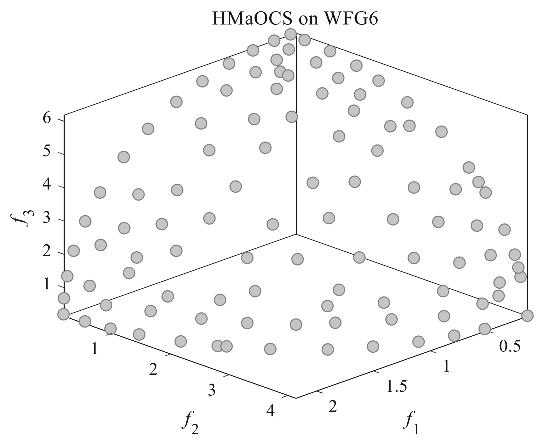
**(f)** Solutions obtained HyPE on DTLZ4

**Fig. 2** Solutions obtained on DTLZ4

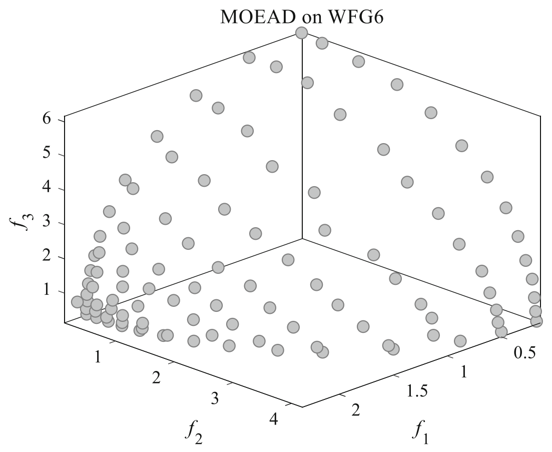
benchmark set. The total results show that NSGA-III is slightly better than HMaOCS. Compared to KnEA, GrEA and HypE, HMaOCS achieves better results on two benchmark sets. Although HMaOCS is slightly worse than NSGA-III, it outperforms MOEA/D, KnEA, CrEA and HypE.

## 6 Conclusion and future work

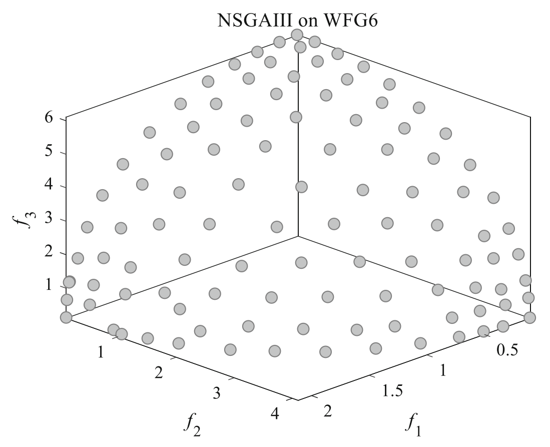
In this paper, we propose a hybrid many-objective cuckoo search (HMaOCS) for MaOPs. The standard CS is only suitable for single-objective optimization problems. To deal with MaOPs, the standard CS is modified. Moreover,



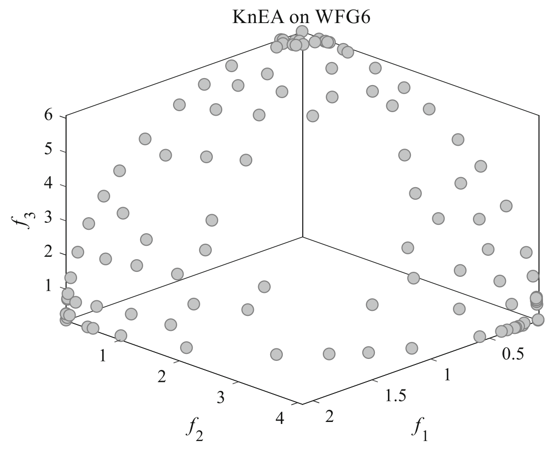
**(a)** Solutions obtained by HMaOCS on WFG6



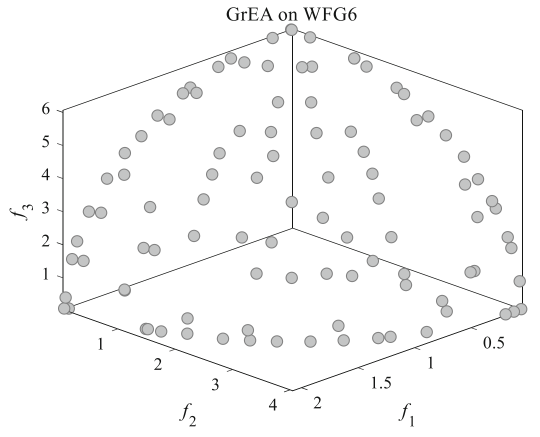
**(b)** Solutions obtained by MOEA/D on WFG6



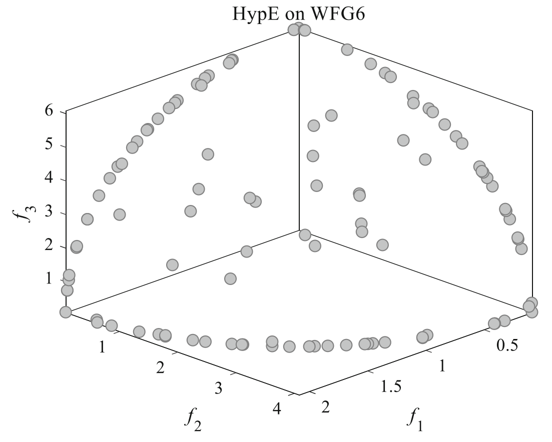
**(c)** Solutions obtained by NSGA-III on WFG6



**(d)** Solutions obtained by KnEA on WFG6



**(e)** Solutions obtained by GrEA on WFG6



**(f)** Solutions obtained by HypE on WFG6

**Fig. 3** Solutions obtained on WFG6

non-dominated sorting and the strategy of references points in NSGA-III are employed to ensure the convergence and diversity. In order to evaluate the performance of HMaOCS, two well-known benchmark sets DTLZ and WFG are utilized in the experiments.

The parameter $p_a$ can affect the performance of HMaOCS. Simulation results show that a fixed $p_a$ is not

**Table 10** IGD results of the six algorithms on WFG1-WFG9

| Problem | *M* | HMaOCS | MOEA/D | NSGA-III | KnEA | GrEA | HypE |
|---|---|---|---|---|---|---|---|
| WFG1 | 2 | 1.0240e+0 (1.95e−1) | 8.4498e−2 (4.46e−2) | 8.0642e−2 (3.40e−2) | 1.6949e−1 (3.02e−2) | **4.8162e−2 (3.04e−2)** | 7.0584e−1 (1.93e−1) |
|  | 4 | 1.2355e+0 (1.59e−1) | 7.6176e−1 (1.35e−2) | **2.8539e−1 (7.82e−3)** | 3.4097e−1 (2.03e−2) | 3.9578e−1 (2.31e−2) | 1.2939e+0 (2.82e−1) |
|  | 6 | 1.9982e+0 (1.43e−1) | 1.5461e+0 (4.72e−2) | 6.5858e−1 (3.26e−2) | **6.5830e−1 (2.40e−2)** | 7.5168e−1 (3.22e−2) | 1.9982e+0 (2.52e−1) |
|  | 8 | 2.3338e+0 (6.72e−2) | 2.1863e+0 (1.08e−1) | **8.5520e−1 (3.22e−2)** | 9.6958e−1 (1.38e−1) | 1.3971e+0 (1.68e−1) | 2.4548e+0 (2.86e−1) |
|  | 10 | 2.7429e+0 (7.71e−2) | 2.4782e+0 (2.03e−1) | **1.0385e+0 (4.81e−2)** | 1.1764e+0 (1.41e−1) | 1.2580e+0 (4.97e−2) | 2.5956e+0 (2.26e−1) |
| WFG2 | 2 | 1.6478e−2 (7.47e−4) | 6.0580e−2 (2.07e−3) | 1.6629e−2 (7.90e−4) | 1.0901e+0 (2.25e−1) | 3.3742e−2 (2.62e−3) | **1.0212e−2 (2.51e−4)** |
|  | 4 | **3.8363e−1 (1.94e−2)** | 2.4835e+0 (5.20e−2) | 3.8930e−1 (2.49e−2) | 4.4246e−1 (5.93e−2) | 6.6525e−1 (7.69e−2) | 6.4223e−1 (1.90e−1) |
|  | 6 | **7.4040e−1 (2.09e−2)** | 5.6806e+0 (5.82e−2) | 9.8052e−1 (1.75e−1) | 7.6260e−1 (7.48e−2) | 1.5644e+0 (3.05e−1) | 1.3906e+0 (2.83e−1) |
|  | 8 | **1.2712e+0 (1.44e−1)** | 8.7708e+0 (5.72e−2) | 2.6043e+0 (1.41e+0) | 1.4411e+0 (1.47e−1) | 2.5494e+0 (6.23e−1) | 2.6895e+0 (4.35e−1) |
|  | 10 | 2.3567e+0 (2.99e−1) | 1.6679e+1 (1.07e−1) | 4.3476e+0 (1.81e+0) | **2.3285e+0 (4.98e−1)** | 2.9514e+0 (2.75e−1) | 5.8419e+0 (7.29e−1) |
| WFG3 | 2 | 1.9722e−2 (4.57e−3) | 2.6386e−2 (5.77e−3) | 1.3891e−2 (9.74e−4) | 1.8005e−2 (8.51e−4) | 2.3433e−2 (5.57e−4) | **1.2256e−2 (4.55e−4)** |
|  | 4 | 5.0105e−1 (5.78e−2) | 5.2869e−1 (1.47e−1) | 2.7844e−1 (3.50e−2) | 3.4010e−1 (5.35e−2) | 2.2282e−1 (1.85e−2) | **4.7390e−2 (3.14e−3)** |
|  | 6 | 1.2453e+0 (1.13e−1) | 1.6849e+0 (4.61e−1) | 9.5047e−1 (1.19e−1) | 8.4309e−1 (1.15e−1) | 6.0824e−1 (1.02e−1) | **7.0825e−2 (6.60e−3)** |
|  | 8 | 2.0352e+0 (2.42e−1) | 3.8407e+0 (1.56e−1) | 1.0677e+0 (3.25e−1) | 1.5551e+0 (4.19e−1) | 1.0070e+0 (1.75e−1) | **8.3688e−2 (9.67e−3)** |
|  | 10 | 2.1031e+0 (4.40e−1) | 5.9602e+0 (4.56e−1) | 1.1013e+0 (3.46e−1) | 2.1324e+0 (5.23e−1) | 1.7794e+0 (2.87e−1) | **9.3512e−2 (9.13e−3)** |
| WFG4 | 2 | 2.2582e−2 (1.90e−3) | 3.4822e−2 (2.11e−3) | **1.3905e−2 (5.93e−4)** | 2.3676e−2 (3.07e−3) | 2.5929e−2 (1.55e−3) | 1.8478e−2 (1.32e−3) |
|  | 4 | 6.1477e−1 (2.97e−3) | 6.5147e−1 (2.92e−3) | **6.0778e−1 (1.39e−3)** | 6.5857e−1 (1.28e−2) | 6.3331e−1 (9.45e−3) | 8.4653e−1 (3.79e−2) |
|  | 6 | **1.7354e+0 (9.43e−3)** | 3.7931e+0 (1.28e−1) | 1.7672e+0 (8.74e−2) | 1.9127e+0 (3.14e−2) | **1.6866e+0 (8.56e−3)** | 2.6639e+0 (4.82e−1) |
|  | 8 | **2.9781e+0 (1.29e−2)** | 7.0973e+0 (9.90e−2) | 2.9793e+0 (1.16e−2) | 3.3672e+0 (2.32e−2) | 2.9889e+0 (1.32e−2) | 5.2701e+0 (6.49e−1) |
|  | 10 | 4.5081e+0 (2.11e−2) | 9.4233e+0 (6.31e−2) | 4.5199e+0 (2.54e−2) | 4.4883e+0 (3.59e−2) | **4.1190e+0 (4.50e−2)** | 8.1356e+0 (9.09e−1) |
| WFG5 | 2 | **6.4390e−2 (3.30e−3)** | 7.2468e−2 (3.19e−3) | 6.5113e−2 (2.33e−3) | 7.4018e−2 (4.62e−3) | 7.4621e−2 (2.38e−3) | 6.7454e−2 (2.08e−3) |
|  | 4 | **5.9717e−1 (2.50e−3)** | 6.6728e−1 (3.56e−2) | 6.0261e−1 (7.25e−4) | 6.6513e−1 (1.91e−2) | 6.2887e−1 (1.16e−2) | 8.7554e−1 (2.04e−2) |
|  | 6 | **1.6989e+0 (4.50e−3)** | 3.4495e+0 (2.26e−1) | 1.7071e+0 (5.14e−3) | 1.8641e+0 (2.96e−2) | 1.7144e+0 (1.33e−2) | 2.1262e+0 (3.69e−2) |
|  | 8 | **2.9844e+0 (1.12e−1)** | 6.6753e+0 (1.64e−1) | 2.9896e+0 (4.16e−3) | 3.3332e+0 (3.53e−2) | 2.9921e+0 (1.46e−2) | 3.4982e+0 (7.56e−2) |
|  | 10 | 4.3307e+0 (1.03e−1) | 9.0123e+0 (1.87e−1) | 4.4769e+0 (1.20e−2) | 4.4566e+0 (3.61e−2) | 4.0576e+0 (2.55e−2) | 5.4168e+0 (4.71e−1) |
| WFG6 | 2 | **5.6757e−2 (2.03e−2)** | 1.2780e−1 (2.44e−2) | 7.3999e−2 (1.81e−2) | 2.8496e−1 (5.34e−2) | 8.8129e−2 (2.43e−2) | 7.5416e−2 (2.08e−2) |
|  | 4 | 6.5997e−1 (1.08e−2) | 7.6857e−1 (6.61e−2) | **6.1379e−1 (3.89e−3)** | 6.8780e−1 (1.68e−2) | 6.6048e−1 (1.23e−2) | 8.9655e−1 (2.40e−2) |

**Table 10** (continued)

| Problem | *M* | HMaOCS | MOEA/D | NSGA-III | KnEA | GrEA | HypE |
|---|---|---|---|---|---|---|---|
| | 6 | **1.7499e+0 (1.06e−2)** | 3.9283e+0 (5.14e−2) | 1.7545e+0 (8.22e−3) | 1.9476e+0 (3.86e−2) | 1.7543e+0 (1.34e−2) | 2.1800e+0 (7.77e−2) |
| | 8 | **2.9788e+0 (1.02e−2)** | 7.2267e+0 (2.02e−1) | 2.9789e+0 (4.59e−3) | 3.4512e+0 (6.98e−2) | 2.9850e+0 (2.90e−2) | 3.6572e+0 (1.29e−1) |
| | 10 | 4.5703e+0 (1.47e−2) | 9.5154e+0 (1.70e−1) | 4.5878e+0 (1.84e−2) | 4.6514e+0 (5.60e−2) | **4.0782e+0 (2.51e−2)** | 5.4771e+0 (5.19e−1) |
| WFG7 | 2 | 1.7696e−2 (7.12e−4) | 3.6122e−2 (6.70e−3) | **1.2688e−2 (1.69e−4)** | 1.2479e−1 (4.78e−2) | 3.0082e−2 (1.19e−3) | 1.7207e−2 (9.71e−4) |
| | 4 | 6.4787e−1 (5.85e−3) | 7.5605e−1 (5.03e−2) | **6.0798e−1 (1.09e−3)** | 6.5618e−1 (1.31e−2) | 6.4884e−1 (1.50e−2) | 9.2472e−1 (2.49e−2) |
| | 6 | **1.7601e+0 (1.22e−2)** | 3.9485e+0 (6.28e−2) | 1.7681e+0 (8.51e−3) | 1.9411e+0 (3.83e−2) | 1.7767e+0 (1.56e−2) | 2.1925e+0 (7.64e−2) |
| | 8 | 3.0408e+0 (1.80e−2) | 7.1663e+0 (8.02e−2) | **2.9819e+0 (9.17e−3)** | 3.3322e+0 (3.72e−2) | 2.9045e+0 (1.35e−2) | 4.2741e+0 (5.15e−1) |
| | 10 | 4.5159e+0 (2.97e−2) | 9.3989e+0 (9.44e−2) | 4.5458e+0 (3.15e−2) | 4.3935e+0 (5.33e−2) | **4.1022e+0 (4.77e−2)** | 6.3865e+0 (5.48e−1) |
| WFG8 | 2 | 1.1910e−1 (8.06e−3) | 1.3063e−1 (5.83e−3) | 1.1998e−1 (2.35e−3) | 4.5644e−1 (4.21e−2) | 1.1244e−1 (9.32e−4) | **1.1012e−1 (2.96e−3)** |
| | 4 | 6.8859e−1 (1.21e−2) | 6.8082e−1 (1.28e−2) | **6.4928e−1 (8.68e−3)** | 7.2370e−1 (1.07e−2) | 6.5783e−1 (8.86e−3) | 8.4682e−1 (1.67e−2) |
| | 6 | 1.7891e+0 (4.67e−3) | 3.4531e+0 (2.28e−1) | **1.7363e+0 (8.56e−3)** | 1.9084e+0 (2.15e−2) | 1.7560e+0 (1.39e−2) | 2.2225e+0 (1.89e−1) |
| | 8 | 3.2093e+0 (4.01e−2) | 6.3694e+0 (1.57e−1) | 3.3757e+0 (2.83e−1) | 3.4992e+0 (7.31e−2) | **3.0963e+0 (4.99e−2)** | 4.3643e+0 (3.46e−1) |
| | 10 | **4.3695e+0 (5.53e−2)** | 8.6377e+0 (1.96e−1) | 5.0225e+0 (5.36e−1) | 4.6158e+0 (3.29e−2) | 5.2085e+0 (4.75e−2) | 6.3041e+0 (4.52e−1) |
| WFG9 | 2 | 6.5196e−2 (8.50e−2) | 1.0088e−1 (7.54e−2) | 2.2760e−2 (2.09e−3) | 2.8516e−2 (3.99e−3) | 3.1099e−2 (3.41e−3) | **2.0809e−2 (1.78e−3)** |
| | 4 | 6.5663e−1 (1.15e−2) | 6.9411e−1 (4.37e−2) | **5.9412e−1 (1.54e−2)** | 6.0993e−1 (9.05e−3) | 6.1150e−1 (9.24e−3) | 8.8423e−1 (3.03e−2) |
| | 6 | 1.7055e+0 (2.14e−2) | 3.6584e+0 (1.62e−1) | 1.7174e+0 (2.00e−2) | 1.7821e+0 (2.87e−2) | **1.6542e+0 (9.41e−3)** | 2.0632e+0 (6.05e−2) |
| | 8 | 3.1495e+0 (1.55e−1) | 6.7568e+0 (1.78e−1) | 2.9515e+0 (2.39e−2) | 3.2253e+0 (1.71e−2) | **2.9098e+0 (1.83e−2)** | 3.5928e+0 (2.62e−1) |
| | 10 | 4.6578e+0 (1.73e−1) | 8.9979e+0 (2.12e−1) | 4.3430e+0 (8.27e−2) | 4.2614e+0 (5.33e−2) | **4.1410e+0 (2.71e−2)** | 6.0198e+0 (5.80e−1) |
| ***w/l/t*** | | | **39/6/0** | **23/22/0** | **30/15/0** | **23/22/0** | **32/12/1** |

**Table 11** Comparison results between HMaOCS and five other algorithms on the DTLZ and WFG benchmark sets

| HMaOCS vs. | MOEA/D | NSGA-III | KnEA | GrEA | HypE |
|---|---|---|---|---|---|
| | *w/l/t* | *w/l/t* | *w/l/t* | *w/l/t* | *w/l/t* |
| DTLZ | 15/20/0 | 16/19/0 | **21/14/0** | **19/16/0** | **25/10/0** |
| WFG | **39/6/0** | **23/22/0** | **30/15/0** | **23/22/0** | **32/12/1** |
| Total | **54/26/0** | 39/41/0 | **51/29/0** | **42/38/0** | **57/22/1** |

suitable for all test instances. $p_a = 0.7$ is the relative best choice for the test suite. Compared to other famous many-objective optimization algorithms, HMaOCS can achieve promising performance. MOEA/D performs better than HMaOCS on the DTLZ benchmark set, while HMaOCS is much better than MOEA/D on the WFG benchmark set. For two benchmark sets, both NSGA-III and HMaOCS obtain similar performance. HMaOCS outperforms KnEA, GrEA and HypE on the DTLZ and WFG benchmark sets.

**Acknowledgements** This study is funded by the National Natural Science Foundation of China under Grant Nos. 61806138, U1636220, 61663028, 71771176, 51775385, 61703279 and 71371142, Natural Science Foundation of Shanxi Province under Grant No. 201801D121127, PhD Research Startup Foundation of Taiyuan University of Science and Technology under Grant No. 20182002, the Distinguished Young Talents Plan of Jiang-xi Province under Grant No. 20171BCB23075, the Natural Science Foundation of Jiang-xi Province under Grant No. 20171BAB202035.

## Compliance with ethical standards

**Conflict of interest** Author Zhihua Cui declares that he has no conflict of interest. Author Maoqing Zhang declares that he has no conflict of interest. Author Hui Wang declares that he has no conflict of interest. Author Xingjuan Cai declares that she has no conflict of interest. Author Wensheng Zhang declares that he has no conflict of interest.

**Ethical approval** This article does not contain any studies with human participants or animals performed by any of the authors.

## References

Abdel-Baset M, Zhou Y, Ismail M (2018) An improved cuckoo search algorithm for integer programming problems. Int J Comput Sci Math 9(1):66–81

Adra S, Fleming P (2011) Diversity management in evolutionary many-objective optimization. IEEE Trans Evol Comput 15(2):183–195

Bader J, Zitzler E (2011) HypE: an algorithm for fast hypervolume-based many-objective optimization. Evol Comput 19(1):45–76

Barthelemy P, Bertolotti J, Wiersma D (2008) A Lévy flight for light. Nature 453(7194):495

Cai X, Gao X, Xue Y (2016) Improved bat algorithm with optimal forage strategy and random disturbance strategy. Int J Bio-inspired Comput 8(4):205–214

Cai X, Wang H, Cui Z, Cai J, Xue Y, Wang L (2018) Bat algorithm with triangle-flipping strategy for numerical optimization. Int J Mach Learn Cybernet 9(2):199–215

Chandrasekaran K, Simon S (2012) Multi-objective scheduling problem: hybrid approach using fuzzy assisted cuckoo search algorithm. Swarm Evol Comput 5:1–16

Coelho L, Guerra F, Batistela N (2013) Multiobjective cuckoo search algorithm based on duffing's oscillator applied to jiles-atherton vector hysteresis parameters estimation. IEEE Trans Magn 49(5):1745–1748

Cortés P, Muñuzuri J, Onieva L, Guadix J (2018) A discrete particle swarm optimisation algorithm to operate distributed energy generation networks efficiently. Int J Bio-Inspired Comput 12(4):226–235

Cui Z, Cao Y, Cai X, Cai J, Chen J (2017a) Optimal LEACH protocol with modified bat algorithm for big data sensing systems in internet of things. J Parallel Distrib Comput 10:1–12. https://doi.org/10.1016/j.jpdc.2017.12.014

Cui Z, Sun B, Wang G, Xue Y, Chen J (2017b) A novel oriented cuckoo search algorithm to improve DV-Hop performance for cyber-physical systems. J Parallel Distrib Comput 103:42–52

Cui Z, Xue F, Cai X, Cao Y, Wang G, Chen J (2018) Detection of malicious code variants based on deep learning. IEEE Trans Industr Inf 14(7):3187–3196

Cui Z, Du L, Wang P, Cai X, Zhang W (2019a) Malicious code detection based on CNNs and multi-objective algorithm. J Parallel Distrib Comput 129:50–58

Cui Z, Li F, Zhang W (2019b) Bat algorithm with principal component analysis. Int J Mach Learn Cybernet 10(3):603–622

Cui Z, Zhang J, Wang Y, Cao Y, Cai X, Zhang W, Chen J (2019c) A pigeon-inspired optimization algorithm for many-objective optimization problems. Sci China Inf Sci 62(7):070212. https://doi.org/10.1007/s11432-018-9729-5

Das I, Dennis J (2006) Normal-Boundary Intersection: a new method for generating the Pareto surface in nonlinear multicriteria optimization problems. SIAM J Optim 8(3):631–657

Deb K, Jain H (2014) An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: solving problems with box constraints. IEEE Trans Evol Comput 18(4):577–601

Deb K, Kalyanmoy D (2001) Multi-objective optimization using evolutionary algorithms. Wiley, New York

Deb K, Thiele L, Laumanns M, Zitzler E (2002a) Scalable multi-objective optimization test problems. In: Proceedings of the 2002 congress on IEEE evolutionary computation. CEC '02, pp 825–830

Deb K, Pratap A, Agarwal S, Meyarivan T (2002b) A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2):182–197

Deb K, Mohan M, Mishra S (2005) Evaluating the ε-domination based multi-objective evolutionary algorithm for a quick computation of Pareto-optimal solutions. Evol Comput 13(4):501–525

Fan J, Li Y, Tang L, Wu G (2018) RoughPSO: rough set-based particle swarm optimization. Int J Bio-Inspired Comput 12(4):245–253

Hanoun S, Nahavandi S, Creighton D, Kull H (2012) Solving a multiobjective job shop scheduling problem using Pareto archived cuckoo search. Emerg Technol Factory Autom IEEE 43:1–8

Huband S, Hingston P, Barone L, While L (2006) A review of multiobjective test problems and a scalable test problem toolkit. IEEE Trans Evol Comput 10(5):477–506

Hughes E (2003) Multiple single objective Pareto sampling. In: The 2003 congress on IEEE evolutionary computation (CEC), vol 4, pp 2678–2684

Jain H, Deb K (2014) An Evolutionary Many-objective optimization algorithm using reference-point based nondominated sorting approach, part II: handling constraints and extending to an adaptive approach. IEEE Trans Evol Comput 18(4):602–622

Laumanns M, Thiele L, Deb K, Zitzler E (2002) Combining convergence and diversity in evolutionary multi- objective optimization. MIT Press, Cambridge, pp 263–282

Li M, Zheng J (2009) Spread assessment for evolutionary multi-objective optimization. In: International conference on evolutionary multi-criterion optimization, Springer, pp 216–230

Li M, Zheng J, Li K, Yuan Q, Shen R (2010) Enhancing diversity for average ranking method in evolutionary many-objective optimization. In: Parallel problem solving from nature, PPSN XI. Springer, Berlin, pp. 647–656

Li M, Yang S, Liu X (2014) Shift-based density estimation for Pareto-based algorithms in many-objective optimization. IEEE Trans Evol Comput 18(3):348–365

Niu Y, Tian Z, Zhang M, Cai X, Li J (2018) Adaptive two-SVM multi-objective cuckoo search algorithm for software defect prediction. Int J Comput Sci Math 9(6):547–554

Pandey H, Chaudhary A, Mehrotra D (2018) Bit mask-oriented genetic algorithm for grammatical inference and premature convergence. Int J Bio-Inspired Comput 12(1):54–69
Pooja P, Chaturvedi P, Kumar P, Tomar A (2018) A novel differential evolution approach for constraint optimization. Int J Bio-Inspired Comput 12(4):254–265
Raja B, Jhala R, Patel V (2017) Many-objective optimization of cross-flow plate-fin heat exchanger. Int J Therm Sci 118:320–339
Rani K, Malek M, Neoh S (2013) Hybrid multiobjective optimization using modified cuckoo search algorithm in linear array synthesis. In: IEEE antennas and propagation conference, pp 1–4
Reynolds AM, Frye MA (2007) Free-flight odor tracking in Drosophila is consistent with an optimal intermittent scale-free search. PLoS ONE 2(4):e354
Shan X, Ye B, Zhang L (2018) Analysis of flow field of hydrodynamic suspension polishing disk based on multi-fractal method. Int J Comput Sci Math 9(1):13–20
Sun B, Cui Z, Dai C (2014) DV-hop localization algorithm with cuckoo search. Sensor Lett 12(2):444–447
Tozer B, Mazzuchi T, Sarkani S (2017) Many-objective stochastic path finding using reinforcement learning. Expert Syst Appl. https://doi.org/10.1016/j.eswa.2016.10.045
Wang Z, Li Y (2015) Irreversibility analysis for optimization design of plate fin heat exchangers using a multi-objective cuckoo search algorithm. Energy Convers Manag 101:126–135
Wang Q, Liu S, Wang H (2012) Multi-objective cuckoo search for the optimal design of water distribution systems. In: International conference on civil engineering and urban planning. https://doi.org/10.1061/9780784412435.072
Wang H, Wang W, Zhou X, Sun H, Zhao J, Yu X, Cui Z (2017) Firefly algorithm with neighborhood attraction. Inf Sci 382(383):374–387
Wang H, Wang W, Cui Z, Zhou X, Zhao J, Li Y (2018) A new dynamic firefly algorithm for demand estimation of water resources. Inf Sci 438:95–106
Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82
Yang X, Deb S (2010a) Cuckoo search via Lévy flights. In: Nature and biologically inspired computing 2009, NaBIC 2009, world congress on IEEE, pp 210–214
Yang X, Deb S (2010b) Engineering optimisation by cuckoo search. Int J Math Model Numer Optim 1(4):330–343
Yang S, Li M, Liu X, Zheng J (2013) A grid-based evolutionary algorithm for many-objective optimization. IEEE Trans Evol Comput 17(5):721–736
Yigit T, Unsal O, Deperlioglu O (2018) Using the metaheuristic methods for real-time optimisation of dynamic school bus routing problem and an application. Int J Bio-Inspired Comput 11(2):123–133
Zhang QF, Li H (2007) MOEA/D: a multiobjective evolutionary algorithm based on decomposition. IEEE Trans Evol Comput 11(6):712–731
Zhang X, Tian Y, Jin Y (2015) A knee point-driven evolutionary algorithm for many-objective optimization. IEEE Trans Evol Comput 19(6):761–776
Zhang M, Wang H, Cui Z, Chen J (2018) Hybrid multi-objective cuckoo search with dynamical local search. Memet Comput 10(2):199–208
Zhao B, Xue Y, Xu B, Ma T, Liu J (2018) Multi-objective classification based on NSGA-II. Int J Comput Sci Math 9(6):539–546
Zhou X, Liu Y, Li B (2016) A multi-objective discrete cuckoo search algorithm with local search for community detection in complex networks. Mod Phys Lett B 30(07):1650080
Zitzler E, Kunzli S (2004) Indicator-based selection in multi objective search. In: Lecture notes in computing science, pp 832–842
Zitzler E, Laumanns M, Thiele L (2001) SPEA2: improving the strength Pareto evolutionary algorithm for multiobjective optimization. In: Giannakoglou K, et al. (eds) EUROGEN 2001. International Center for Numerical Methods in engineering (CIMNE), pp 95–100
Zou X, Chen Y, Liu M, Kang L (2008) A new evolutionary algorithm for solving many-objective optimization problems. IEEE Trans Syst Man Cybern B Cybern 38(5):1402–1412