Appl Intell
DOI 10.1007/s10489-017-0927-y

Multi-objective hybrid artificial bee colony algorithm enhanced with Lévy flight and self-adaption for cloud manufacturing service composition

Jiajun Zhou[1] · **Xifan Yao**[1]

Abstract Service composition and optimal selection (SCOS) is a key problem in cloud manufacturing (CMfg). The present study proposed a multi-objective hybrid artificial bee colony (HABC) algorithm to address the SCOS problem in consideration of both quality of service (QoS) and energy consumption, to which an improved solution update equation with multiple dimensions of perturbation was adopted in the employed bee phase. Likewise, a cuckoo search-inspired Lévy flight was employed in the onlooker bee phase to overcome basic artificial bee colony (ABC) drawbacks such as poor exploitation and slow convergence. Moreover, a parameter adaptive strategy was applied to adjust the perturbation rate and step size of the Lévy flight to improve the performance of the algorithm. The proposed algorithm was first tested on 21 multi-objective benchmark problems and compared with four other state-of-the-art multi-objective evolutionary algorithms (MOEAs). The effect of the improvement strategies was then experimentally verified. Finally, the HABC was applied to solve multiscale SCOS problems using comparison experiments, which resulted in more competitive results and outperformed other MOEAs.

Keywords Cloud manufacturing · Multi-objective optimization · Lévy flight · Artificial bee colony algorithm · Service composition

✉ Xifan Yao
mexfyao@scut.edu.cn

Jiajun Zhou
jiajunzhou163@163.com

[1] School of Mechanical and Automotive Engineering, South China University of Technology, Guangzhou, 510640, Guangdong, China

1 Introduction

Information technology advances bring unprecedented opportunities for many industries, especially in the recent rise of several cutting-edge technologies such as cloud computing, cyber-physical systems and the Internet of Things (IoT). The increased integration of manufacturing things (resources) with sensors in the cloud-based Internet preceded the emergence of cloud manufacturing (CMfg) [1]. Well-established connections between various manufacturing resources and capabilities (MR&Cs) and the Internet [2] assist CMfg in providing users with full sharing and on-demand use of MR&Cs in service form to change the way enterprises execute business plans. As such, CMfg is predicted to alter the manufacturing industry by means of improved perception, faster planning, more accurate executions, and user self-provisioning based on the pay-as-you-go principle and other techniques.

CMfg has many issues, e.g. task semantic modeling [3], formal description of manufacturing capability [4], resource virtualization [5] and sharing strategies [6], and service composition and optimal selection (SCOS) [7], wherein SCOS has been deemed a principal challenge. In a CMfg system, distributed MR&Cs are virtualized and encapsulated as cloud services to ease user searches and assembly when executing tasks. The exuberant growth of services in the CMfg resource pool increased the difficulty for the combination of multiple single services into a more capable and powerful composite service that meets the user's requirements while maintaining optimal service performances. This difficulty is one of the most important issues that require further examination. Due to the large-scale characteristic of SCOS, the application of exact approaches or exhaustive search algorithms is limited. Evolutionary algorithms have been developed to solve the problem, such as

Published online: 19 April 2017

chaos optimization [7–9], genetic algorithm (GA) [10, 11], ant colony optimization (ACO) [12], and ABC [13].

Evolutionary algorithms can obtain near-optimal solutions within a short period of time and have become a thriving area of research and academic efforts. Although SCOS is a multi-objective problem (MOP), almost all existing studies are focused only on QoS and have only obtained a single appropriate solution. However, it is more sensible to provide MOP decision-makers with a set of non-dominated solutions. The growing interest in solving sustainability issues has garnered an urgent need to consider the economic, environmental, and social aspects of manufacturing system [14]. In particular, the escalating demand for energy poses serious challenges to sustainable manufacturing [15] and thus requires continuous efforts to produce more sustainable manufactured products [16], which may involve multi-objective tradeoffs between the economic and environmental aspects of SCOS. The present study aims to provide an efficient approach for SCOS in consideration of these economic and environmental aspects.

ABC is one of the most recently developed metaheuristic algorithms based on the intelligent foraging behavior of a honey bee swarm [17]. ABC has been widely used in many fields [18–20] as it is easy to implement and has fewer control parameters [21, 22]. ABC performs a neighborhood search of food locations throughout the whole search process. This characteristic highlights ABC's applicability for good exploration but poor exploitation capabilities, which result in relatively slow convergences of ABC in addressing certain complex issues. Yang and Deb recently proposed another novel evolutionary algorithm, namely the cuckoo search (CS) [23]. CS with Lévy flight performs an exploitative local or neighborhood search combined with occasional long jumps, which allow more efficient search space explorations to escape from the local optima. Comparison studies illustrate a more robust and precise search using CS than by using algorithms such as GA and ABC for complex multi-model optimization problems [24]. However, the convergence speed of the CS is relatively slow. The present study proposes a hybrid MOEA based on the hybridization of ABC and CS to deal with multi-objective SCOS, which ultimately takes advantage of CS and minimizes shortcomings of ABC. In addition, the original design of the onlooker bee's movement only considers a single dimension, which results in insufficient information sharing when handling complex problems. To overcome this limitation, a multi-dimension perturbation mechanism was designed for the employed bees.

The rest of the paper is organized as follows: Section 2 reviews relevant works. In Section 3, SCOS that minimizes energy consumption while simultaneously maximizes QoS is mathematically formulated. In Section 4, HABC details are provided to address SCOS. Section 5 presents the experimental studies on benchmark problems. In Section 6, HABC is applied to solve SCOS. Finally, Section 7 summarizes the research conclusions and discusses future directives.

2 Related works

Although CMfg services may expose many operations, a single service generally acts fairly for atomic, low-level functions and cannot satisfy the various complicated requirements of many real-world cases, particularly in the presence of complex and diverse customer-generated tasks in personalized customization. Thus, service composition is one of the best approaches when encountering these challenges. However, selecting an optimal composition service is an NP-hard problem.

Traditionally, exact methods such as integer programming [25] and mixed integer programming [26] have long been applied to address these issues, though they perform well in some specific situations but fail in more complex situations. SCOS is generally an NP-hard problem in an n-dimensional hyperspace, where the QoS variables can be large and have complex non linear influence on the objective function of SCOS. Identifying the optimal solution using exact methods is computationally expensive and cannot be accomplished within limited time constraints, thereby resulting in the development of evolutionary algorithms to achieve better scalability. Zhang et al. [27] designed an improved GA for enterprise partner selection in consideration of the green criteria. Zhao et al. [28] proposed an improved particle swarm optimization (PSO) for SCOS. Wang et al. [12] constructed a culture max-min ant system (C-MMAS) algorithm to solve SCOS for virtual enterprises. However, research has focused on very limited and popular approaches, such as GA, PSO, and ACO, which have already been widely reported. Minimal efforts have been observed in exploring promising new algorithms for SCOS. To address such issues, the present study will introduce newly developed algorithms such as the cuckoo search (CS) and ABC to address SCOS. The superior performance of CS compared to other well-known algorithms has been demonstrated in [23, 24, 29]. Therefore we attempted to introduce the CS into ABC for solving SCOS.

Although service composition problems have been widely studied, research on service selection and scheduling in cloud system and CMfg is still in its initial stages. Zhang et al. [30] investigated the flexible management of service composition in CMfg. However, the formal description and algorithm for SCOS was not addressed. Wang et al. [11] proposed a cloud service composition method based on GA. In [10], a hybrid approach using GA and fruit fly optimization was introduced to solve SCOS in cloud computing. In [13], the authors adopted a global best-guided

ABC to solve cloud service composition. Laili et al. [9] proposed the ranking-based chaos optimization by simultaneously considering SCOS and the allocation of computing resources. Huang et al. [8] designed chaos optimization with consideration of the energy consumption for cloud service selection. In [7], an adaptive chaos optimization called full connection based parallel adaptive chaos optimization with reflex migration (FC-PACO-RM) was developed to solve SCOS. However, these studies treated multi-objective SCOS problems as single objective optimization ones [8, 13].

MOEAs have garnered attention for simultaneously addressing multiple objectives for MOPs as well as obtaining a set of representative and well-distributed solutions in a single run, specifically with MOEAs such as the Non-dominated Sorting Genetic Algorithm II (NSGA-II), micro-GA, Strength Pareto Evolutionary Algorithm 2 (SPEA2), and multi-objective PSO (MOPSO) [31]. Currently, some typical MOEAs have been applied in MOPs. Ramacher et al. [32] presented an improved NSGA-II to determine the Pareto solutions for service selection. In [33], SPEA2 was introduced to multi-objective SCOS to achieve the Pareto solutions with a well-spread distribution. In [34], set evolution MOPSO was proposed to solve multi-objective problems. In [35], the performance of several classical MOEAs for SCOS was compared. Despite exhibiting good performances in their specified domains, classic MOEAs such as NSGA-II, micro-GA , SPEA2, and MOPSO were extended from the well-known GA and PSO, with many publications have reported on their enhancements and applications. Nevertheless, limited efforts have been pursued on more recently developed algorithms for SCOS, such as multi-objective grey wolf optimizer [36], multi-objective artificial raindrop algorithm [37], multi-objective gravitational search algorithm [38], multi-objective cat swarm optimization [39], multi-objective teaching learning based optimization [40], and multi-objective ABC (MOABC) algorithm [41], some of which truly support MOPs, especially MOABC, which is easily implementable and relatively efficient in comparison with others. Despite its impressive performance in practical applications [19, 41, 42], MOABC has not yet been applied to solve a multi-objective SCOS problem.

The classic ABC has presented some drawbacks, specifically prematurity and slow convergence, in finding near-optimal solutions for complex MOPs. To overcome such drawbacks, a hybrid ABC (HABC) is proposed. In HABC, the multi-dimension perturbation mechanism is employed to improve convergence speed, while the Lévy flight of CS is adopted to avoid prematurity. Lévy flight has an infinite mean and variance, thus it can explore the search space more efficiently as compared to algorithms with random permutation. This advantage, combined with the multidimensional perturbation mechanism designed for employed bees, allows the search space to be explored more efficiently. Furthermore, HABC parameters are dynamically configured based on the previous performances of generating promising solutions.

3 Problem statement

Generally, a complex task in CMfg is decomposed into several subtasks and fulfilled by a composite service. SCOS involves the determination and selection of manufacturing services (MSs) for each subtask such that the obtained composite service satisfies both the functional and QoS requirements. The SCOS process can be divided into following steps: (1) Task decomposition; (2) Service searching and matching; (3) Qualified candidate services set acquisition; (4) Composition of optimal services. More detailed matching and composition flows were investigated in the authors' previous work [43].

Unlike single-objective problems, multi-objective SCOS problems try to simultaneously optimize all objectives by balancing multiple conflict objectives to get a set of non-dominated solutions. Concerns for the SCOS in the present study are focused on both providing the best QoS and reducing energy consumption.

3.1 QoS evaluation

QoS consists of several attributes for evaluating the MSs from different aspects, such as time (T), cost (C), availability (Av), reliability (Re) [7, 44]. The aggregation value of QoS for a composite manufacturing service (CMS) is evaluated based on composite structures (i.e., sequential, parallel, conditional, and loop). The related mathematical model was built in the authors' previous work [45]. The QoS for CMS can be calculated as follows:

$$QoS(CMS) = \sum_{k=1}^{r} w_k \times NormQ_k(CMS) \tag{1}$$

subject to

$$\forall k \in \{1, 2, ...r\} \begin{cases} agg(Q_k) \geq Q_{k0}, \text{ if } Q_k \text{ is a positive attribute} \\ agg(Q_k) \leq Q_{k0}, \text{ if } Q_k \text{ is a negative attribute} \end{cases} \tag{2}$$

where $NormQ_k(CMS)$ represents the normalized value of the kth attribute for CMS, r is the number of attributes, w_k represents the weight of the kth attribute, $w_k \in [0, 1]$, $\sum_{k=1}^{r} w_k = 1$, and Q_{k0} denotes the lowest requirement of the kth attribute.

3.2 Energy consumption evaluation

Although many criteria have been considered in the QoS evaluation of cloud services, "green criterion" such as carbon emission and energy consumption have been minimally considered. However, the increasing public awareness of environmental protection have gradually rendered green considerations as a significant factor in selecting qualified manufacturing service (MS) [8, 27, 46], among which energy consumption has become a significant component in such a way that it has been selected as one of the optimization goals for SCOS in this study.

Traditionally, services are deployed into a fixed computer and require specific maintenance, whereas the cloud services in CMfg have a larger scale and are mainly supported by virtual machines (VMs) [9]. In CMfg, VM-supported cloud services not only include software services (SSs) such as simulation and design software, but also include hardware services (HSs) such as machine tools and manufacturing equipment. The modeling and evaluation of energy consumption for SSs and HSs must be analyzed separately due to their different characteristics [47].

3.2.1 Energy consumption evaluation for SSs

SSs in CMfg are running with the support of VMs, which are the virtual division of the underlying computing resources. The resource information of SS, $\boldsymbol{S}_1$, can be represented as:

$$\boldsymbol{S}_1 = (s, v, p, c, q, f, u, g)$$

where s represents the service execution efficiency in bits per second; v and p denote the minimum required speed and running speed of VM, respectively, in bits per second; c denotes the input communication amount from the predecessor node, in bytes; q denotes the transmission rate of VM, in bits per second; f denotes the failure probability of VM; u denotes the recovery time if VM fails; and g denotes the average energy consumption of VM per unit time.

If the data size of a decomposed subtask is w and the input communication amount is c, then the total energy consumption $En(\boldsymbol{S}_1)$ of the software service can be calculated:

$$En(S_1) = T_1 g = \left(\frac{vw}{ps} + \frac{c}{q} + fu\right) g \tag{3}$$

where T_1 denotes the total execution time.

3.2.2 Energy consumption evaluation for HSs

Unlike SSs, most HSs require supervision or control during execution, thus VMs are deemed indispensable as they are employed to control and supervise HSs. The transmission path of the control commands for SSs is 'user-VM', whereas the transmission path for HSs is 'user-VM-HSs'. Such a process will increase service execution time and produce a large amount of energy consumption [9]. For this reason, more factors must be considered in the energy consumption of HSs and VMs, namely ζ, η and e, that require additional consideration. The resource information of HS, $\boldsymbol{S}_2$, can be represented as

$$\boldsymbol{S}_2 = (s, v, p, c, q, f, u, g, \zeta, \eta, e)$$

where ζ denotes the average control rate, η represents the transferring rate of data between VM and HSs, and e denotes the average energy consumption of HS per unit time.

The other parameters, namely s, v, p, c, q, f, u, and g, have been previously explained. The total energy consumption $En(\boldsymbol{S}_2)$ can be calculated as follows:

$$En(S_2) = T_2(g+e) = \left(\frac{vw(\eta + s\zeta)}{ps\eta} + \frac{c}{q} + fu\right)(g+e) \tag{4}$$

where T_2 denotes the total execution time of HS. If the HS does not require external commands, then $\zeta = 0$.

3.2.3 Objective function of energy consumption

Let $\boldsymbol{X}$ denote the mapping between the subtask and the corresponding service, such that $\boldsymbol{X}(i) = j$ signifies that subtask ST_i has been assigned to service MS_{ij}. For $i \in \{1, 2, ..., n\}$, $\boldsymbol{X}(i) \in \{1, 2, ..., m_i\}$, where n is the number of subtasks and m_i is the number of candidate services for ST_i. Then the objective function is stated as follows:

$$En(CMS) = En_1(CMS) + En_2(CMS) \tag{5}$$

where

$$En_1(CMS) = \sum_{i=1}^{n_{s_1}} \left(\frac{v_i w_i}{p_{i,X(i)} s_{i,X(i)}} + \frac{c_i}{q_{i,X(i)}} + f_{i,X(i)} u \right) g_{i,X(i)}, \quad CMS \in \mathrm{S}_1; \tag{6}$$

$$En_2(\mathrm{CMS}) = \sum_{i=1}^{n_{s_2}} \left(\frac{v_i w_i (\eta + s_{i,X(i)}\zeta)}{p_{i,X(i)} s_{i,X(i)} \eta} + \frac{c_i}{q_{i,X(i)}} + f_{i,X(i)} u \right) (g_{i,X(i)} + e_{i,X(i)}), \; CMS \in \mathrm{S}_2. \tag{7}$$

Here, n_{s1} and n_{s2} are the number of SSs and HSs, respectively; and $En_1(CMS)$ and $En_2(CMS)$ represent the energy consumption of the SSs and HSs, respectively.

3.3 Objective functions of multi-objective SCOS

As previously stated, QoS and energy consumption are considered as two objectives in SCOS, where time (T), cost

(C), availability (Av), and reliability (Re) are treated as constraints. The problem can be then formalized as follows:

$$\text{Min} \quad F(\text{CMS}) = \{1 - QoS(\text{CMS}), En(\text{CMS})\} \tag{8}$$

$$\text{St.} \begin{cases} agg(Q_k) \geq Q_{k0}, & \text{if } Q_k \text{ is a positive attribute} \\ agg(Q_k) \leq Q_{k0}, & \text{if } Q_k \text{ is a negative attribute} \end{cases}$$
$$\forall k \in \{1,\ 2,\ \ldots r\} \tag{9}$$

where the constraint in (9) requires the QoS criteria to satisfy the lowest requirement of each subtask.

4 Multi-objective hybrid ABC algorithm

This section starts with a brief review on Pareto solutions as well as the standard ABC and CS algorithms to propose the HABC using a multi-dimension perturbation mechanism and Lévy flight with adaptive parameter settings.

4.1 Pareto solutions for MOPs

For a minimization MOP, a solution $\boldsymbol{X}_1$ is said to dominate another solution $\boldsymbol{X}_2$ (denoted as $\boldsymbol{X}_1 \prec \boldsymbol{X}_2$), if and only if:

$$f_i(\boldsymbol{X}_1) \leq f_i(\boldsymbol{X}_2), \quad i = 1, 2, \cdots, M, \tag{10}$$

$$f_j(\boldsymbol{X}_1) < f_j(\boldsymbol{X}_2), \quad \exists j \in \{1, 2, \cdots, M\} \tag{11}$$

where f denotes the objective functions and M is the number of objectives. Considering Pareto dominance, a vector $\boldsymbol{X}_0$ is called the Pareto solution if and only if $\neg\exists \boldsymbol{X} \in S$ such that $\boldsymbol{X} \prec \boldsymbol{X}_0$. The Pareto solution set (PS) is defined as follows:

$$\text{PS} = \{\boldsymbol{X}_0 | \neg\exists \boldsymbol{X} \in S \ \text{ and } \ \boldsymbol{X} \prec \boldsymbol{X}_0\} \tag{12}$$

where S denotes the search space. Figure 1 graphically shows an example of Pareto solutions (marked as the black circles) and dominated solutions (marked as the empty circles) for a two-objective optimization problem, in which both objectives require minimization.

4.2 Basic ABC algorithm

Karaboga proposed the ABC algorithm to simulate honey bee behavior during the foraging cycle [17]. The top level pseudocode of ABC is provided in Algorithm 1.

Algorithm 1 ABC top level pseudocode

1: Initialization.
Repeat
2: Send employed bees to exploit potential food sources.
3: Send onlooker bees to exploit promising food sources.
4: Send scout bees to search for new food sources.
5: Memorize the best solution found so far.
Until termination criterion met.

In initialization, the food sources (solutions) of ABC are randomly generated as follows:

$$x_{ij} = x_j^{\min} + rand(0, 1)(x_j^{\max} - x_j^{\min}) \tag{13}$$

where x_{ij} denotes the jth dimension of the ith solution, $[x_j^{\max},\ x_j^{\min}]$ is the boundary of the jth dimension, and $rand$ (0, 1) is a random number uniformly generated within the range [0,1].

An employed bee, i, tries to improve a solution, $\boldsymbol{X}_i = (x_{i1}, x_{i2}, \ldots, x_{ij}, \ldots, x_{id})$, by modifying one dimension as follows:

$$v_{ij} = x_{ij} + \varphi_{ij}(x_{ij} - x_{kj}) \tag{14}$$

where v_{ij} is the jth dimension of $\boldsymbol{V}_i$ and $\boldsymbol{V}_i = (v_{i1}, v_{i2}, \ldots, v_{id})$ is the updated solution; φ_{ij} is a random number within the range $[-1, 1]$; and i, k, and j are randomly chosen indexes with $i, k \in \{1, 2, \ldots, SN\} (i \neq k)$, and $j \in \{1, 2, \ldots, d\}$, where SN is the number of onlookers. The greedy selection is then performed between $\boldsymbol{V}_i$ and $\boldsymbol{X}_i$ to obtain and retain the better solution.

An onlooker bee chooses a solution depending on the probability value, p_i, as follows:

$$p_i = f_i / \sum_{k=1}^{SN} f_k \tag{15}$$

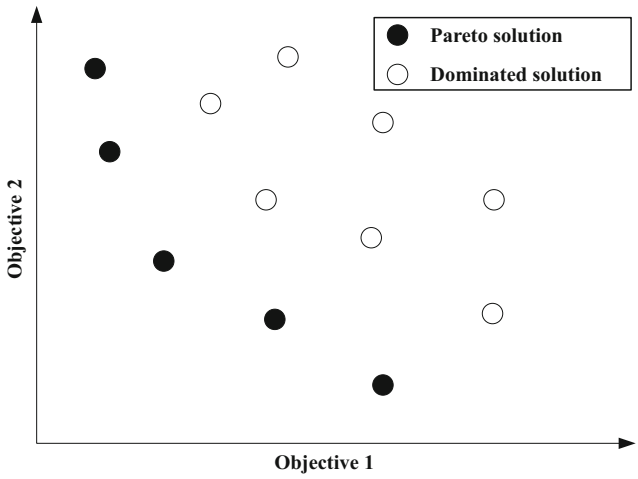

Fig. 1 Illustration of Pareto solutions for a two-objective problem

where f_i represents the fitness of the ith solution, $i \in \{1, 2, \ldots, SN\}$, and SN is the total number of onlooker bees in the population.

If one solution has not improved over a predefined number of times in both the employed bee and onlooker phases, a scout bee generates a new random solution based on (13), which replaces the exhausted one [21, 22].

4.3 Cuckoo search with Lévy flight

Cuckoo search (CS) is one of the latest nature-inspired metaheuristic algorithms. Yang and Deb [23] developed the algorithm in 2009 after gaining inspiration from the obligate brood parasitic behavior of some cuckoo species in combination with the Lévy flight behavior of albatrosses, fruit flies, and spider monkeys. Preliminary studies deem it very promising and far more efficient than existing algorithms such as GA, PSO, and differential evolution (DE) [24, 29, 48].

Each cuckoo in a CS nest represents an existing solution, while a cuckoo egg represents a new solution. The main steps can be summarized as follows:

Given that a new solution, $\boldsymbol{X}_{i,new}$, for cuckoo egg i is generated from a randomly selected cuckoo/solution, $\boldsymbol{X}_i$, a Lévy flight can then be performed as follows:

$$\mathbf{X}_{i,new} = \mathbf{X}_i + \boldsymbol{\alpha} \oplus \text{Levy}(\beta) \tag{16}$$

where the product$\oplus$deals with an entry-wise multiplications process, and α is the step size parameter related to the scale of the problem, which can be expressed as follows:

$$\boldsymbol{\alpha} = SF(\mathbf{X}_i - \mathbf{X}_{best}) \tag{17}$$

where $\boldsymbol{X}_{best}$ is the current best solution. For MOPs, $\boldsymbol{X}_{best}$ can be a non-dominated solution that is randomly selected from external Pareto archive sets, and the scaling factor (SF) is a constant. Levy(β) obeys the Lévy distribution as follows:

$$\text{Levy}(\beta) \sim u = t^{-1-\beta}, \quad 0 < \beta \leq 2 \tag{18}$$

where $\beta \in (0, 2]$ defines the index and determines the shape of the distribution, and t denotes the step length. For the convenience of calculation, a simplified scheme of Levy(β) can be written as follows:

$$\text{Levy}(\beta) \sim \frac{\phi \times u}{|v|^{1/\beta}} \tag{19}$$

where u and v obey the standard normal distributions as follows:

$$u \sim N(0,\ 1) \quad \text{and} \quad v \sim N(0,\ 1) \tag{20}$$

and

$$\phi = \left\{ \frac{\Gamma(1+\beta) \times \sin(\pi \times \beta/2)}{\Gamma[(1+\beta)/2] \times \beta \times 2^{(\beta-1)/2}} \right\}^{1/\beta} \tag{21}$$

where Γ is the standard Gamma function as calculated below:

$$\Gamma(1+\beta) = \int_0^\infty t^\beta e^{-t} dt \tag{22}$$

Mantegna [49] suggested specific β values, i.e., $0.75 \leqslant \beta \leqslant 1.95$, to create a faster and more efficient algorithm. The value 1.5 was determined most suitable among the different β values. According to (18), (19), (20), (21), and (22), a new solution with a Lévy distribution can be calculated as follows:

$$\mathbf{X}_{i,new} = \mathbf{X}_i + SF \frac{\phi \times u}{|v|^{1/\beta}} (\mathbf{X}_i - \mathbf{X}_{best}) \tag{23}$$

Lévy flights consist majorly of small steps and occasionally large steps or long-distance jumps. Such long jumps can significantly improve the search performance of the cuckoo search for some cases, especially in a large-scale irregular solution space.

4.4 Proposed hybrid ABC algorithm to address multi-objective SCOS problems

A hybrid Pareto-based ABC algorithm for solving the SCOS is proposed, wherein the global search ability of ABC and the exploitation ability of Lévy flight is combined. Firstly, the multi-dimension perturbation mechanism is introduced in the employed bee phase of ABC for more efficient global search space explorations. The cuckoo search with Lévy flight is then incorporated in the onlooker bee phase of ABC to refine the exploitation. In addition, the control parameters are gradually self-adapted by learning from their previous experiences in generating promising solutions.

4.4.1 Solution encoding

The integer coding method is applied to encode the candidate services and map the composition services into the position vectors, where the integer value of each vector dimension represents the index of a concrete service from the corresponding candidate set, as depicted in Fig. 2. It is assumed that a task consists of n subtasks. Under the integer array coding scheme, the position vector of a bee is denoted by an n-dimension array, $\boldsymbol{X}_i = \{x_{i1}, x_{i2}, ..., x_{in}\}$, wherein each element x_{ij} denotes the candidate service, $\text{MS}_{i,j}$. For example, the corresponding array of the position vector shown in Fig. 2 is $\{2, 1, ..., m\}$.

4.4.2 External Pareto archive set

HABC uses an external Pareto archive set which acts as an elite archive to store non-dominated solutions found during the search process, unlike the general single-objective

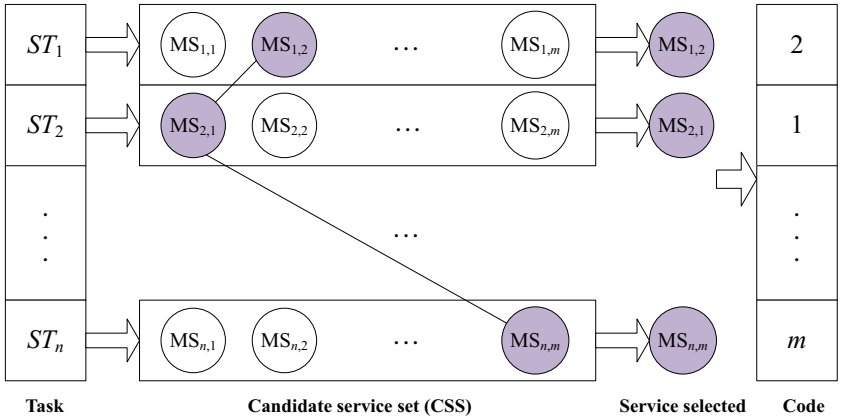

Fig. 2 Array integer encoding of solution

optimization process. At the end of each iteration, the non-dominated solutions are stored in the set and the dominated members in the set are deleted. The external archive set is updated at each iteration.

4.4.3 Initialization

A random set of feasible solutions is generated to initialize food sources, to which non-dominated solutions are added. The initialization procedure for food sources is depicted in Algorithm 2. First, a set of integer solutions is randomly generated within the range of [1, m], where m denotes the number of candidate services for each subtask. The solutions are then judged on their feasibility, which requires the satisfaction of the constraints in (9). Otherwise, the solution will be regenerated. In this way, the selected initial food sources (solutions) will have better performances [27].

Algorithm 2 Pseudocode for food source initialization

Repeat
1: Randomly generate a set of solutions.
2: If it is not feasible, go to Step 1.
3: Add the non-dominated solutions to the food sources.
Until (the *SN* food sources are obtained).

4.4.4 Employed bees with multi-dimension parameter perturbation

In the employed bee phase of ABC, an employed forager probabilistically produces a modification on the food source position (solution) with a single dimension to generate a new solution, which may result in a slow convergence speed and poor exploitation ability in the ABC algorithm when working with complicated composite and non-separable functions. In order to overcome the drawback caused by the single-dimension parameter perturbation, a perturbation rate (*PR*) that controls the perturbation frequency is introduced to the solution-updating equation such that multiple dimensions of a food source are changed at each iteration in the employed bee stage.

By means of such a modification, a new candidate food source is generated for each food source, x_{ij}, as follows:

$$v_{ij} = \begin{cases} x_{ij} + \varphi_{ij}(x_{ij} - x_{kj}), & \text{if } r_{ij} < PR \\ v_{ij}, & \text{otherwise} \end{cases} \tag{24}$$

where r_{ij} is a random number uniformly generated within the range [0, 1]; $k \in \{1, 2, \ldots, SN\}$is a randomly chosen index and $k \neq i$; the dimension $j \in \{1, \ldots, n\}$ is uniformly selected at random; and *PR* is the perturbation rate with a value between 0 and 1. A lower *PR* value results in a slow convergence rate, whereas an extremely high value may cause too much diversity in a population.

4.4.5 Probability calculation

In the basic ABC, solutions are assigned fitness values and onlookers select the individuals for mutation based on the probability in (15). However, in the multi-objective ABC version, the fitness function is not available given the existence of a set of non-dominated solutions. For this reason, a new metric fitness calculation method is introduced to compute the fitness of an individual that is selected by the onlookers for exploitation as follows:

$$fit(\boldsymbol{X}_i) = (2^{R(\boldsymbol{X}_i)} + \tfrac{1}{1+de(\boldsymbol{X}_i)})^{-1} \tag{25}$$

where $R(\boldsymbol{X}_i)$ is the Pareto rank value of individual $\boldsymbol{X}_i$, and $de(\boldsymbol{X}_i)$ is the crowding entropy that is calculated by the fast non-dominated sorting crowding distance measure method [50] and the distribution entropy technique [51]. The combination of the crowding distance and the distribution entropy can exactly reflect the crowding degree of a solution in the objective function space. As such, individuals are selected based on their ranks and crowding entropies. An individual with a lower rank and higher crowding entropy is more desirable.

4.4.6 The onlooker bee phase improved by cuckoo search with Lévy flight

In ABC, the solution search equation for onlookers is similar to that of employed bees. An onlooker updates its position using the information provided by a randomly selected potential solution, therefore the obtained solution may be significantly influenced by a random selected solution. Meanwhile, a forager flies at random between the position itself and a randomly selected solution, with the updated solution depending highly on the step size. If the step size is large, for example, in the case where the difference between the current and randomly selected solution is large with a high absolute value of φ_{ij}, then there is a higher likelihood for onlookers to skip the true solution. On the other hand, if the step size is small, the convergence speed of ABC may significantly decrease.

In order to overcome such a drawback, the cuckoo search-inspired Lévy flight is incorporated into the onlooker bee search phase. Lévy flight uses Lévy distributions instead of uniform and/or Gaussian distributions as mechanisms to generate step sizes. Frequent short steps generated by Lévy distributions intensifies the exploitative local or neighborhood search around the current promising food sources more precisely, which helps the colony search for food sources more quickly and efficiently and enhances the exploitation capability of ABC. Meanwhile, occasional Lévy distribution-produced long jumps explore very different areas of the current search space to scout for potential solutions, which helps foragers escape from the local optima using an exploratory global search.

Lévy flight motions are considered more efficient than stochastic search, especially with no prior knowledge on the locations of food sources, because Lévy flight paths exhibit the characteristics of a series of scale-free moves and Lévy distributions. Various empirical and theoretical studies have validated the characteristics of these Lévy flight patterns [52]. Equation (18) illustrate the scale-free Lévy flights, which consist of sequences of independent and randomly oriented steps that obey an inverse power law distribution with a heavy and long tail. Lévy movements are comprised of many short moves that are punctuated by rare longer moves, which can be used as the optimal local search for onlooker bees. Thus, Lévy flight was chosen as the local search strategy to closely examine the surroundings of some promising regions.

It is important to note the trade-off between the global and local searches for Lévy flights, given that the step length of Lévy flights is controlled by a scaling factor (*SF*). A lower value of *SF* allows onlooker bees to finely tune the search process using smaller steps at a slower convergence rate. A higher value of *SF* accelerates the search process and presents the global random walk, but reduces the exploitation capability of the perturbation process. Thus, we tuned *SF* with the adaptive parameter configuration strategy described in the following section.

4.4.7 Self-adaptation of PR and SF

As mentioned earlier, *PR* and *SF* control the perturbation rate of the food sources and step length of the Lévy flight, respectively. The control parameters, *PR* and *SF*, have a great impact on the exploration/exploitation abilities of the ABC search process. *PR* decides the difference between the parent solution and the generated trial solution and it largely impacts the coverage speed of the algorithm. A larger *PR* value promotes the perturbation of food sources such that the algorithm can explore the search space to find more promising new solutions. A lower *PR* value favors the local search, thereby improving the exploitation performance. Similarly, a high *SF* value is devoted to exploring new solutions, thus inducing a raise in the population diversity, whereas a small *SF* value favors short-distance exploitation and enhances the local exploitation ability of the algorithm. For this reason, both *PR* and *SF* should be adapted to particular problems or particular phases of a search process. The present study proposed a novel adaptive dynamic parameter configuration mechanism to determine the best values of *PR* and *SF* based on previous experiences so as to balance the exploitative local search and exploratory global search during the evolution process, which are described as follows.

Firstly, a current parameter configuration list (CPCL) with a specified length (Fig. 3a) is generated to reserve the parameter sets of *PR* and *SF*, both of which are randomly generated within a uniform distribution range [0, 1]. A parameter set is then selected from the CPCL and assigned to the solution-updating equation of the employed bees and onlookers, respectively. If the updated solution replaces the old one by using the selected parameter set and enters into the next generation, the associated parameter set is then added into a successful parameter configuration list (SPCL) (Fig. 3b) and cleared away from the CPCL.

CPCL		
Index	*PR*	*SF*
1	PR_1	SF_1
2	PR_2	SF_2
...	...	...
L	PR_L	SF_L

(a) Current parameter configuration list

SPCL		
Index	*PR*	*SF*
1	PR_{S1}	SF_{S1}
2	PR_{S2}	SF_{S2}
...	...	...
L	PR_{SL}	SF_{SL}

(b) Successful parameter configuration list

Fig. 3 Parameter configuration list

Once the CPCL was empty, the *PR* and *SF* median values stored in the SPCL were calculated and denoted as m*PR* and m*SF*, respectively. The CPCL was then refilled with the values of *PR* and *SF* that were randomly generated based on the normal distributions, namely N(m*PR*, 0.1) and N(m*SF*, 0.1), respectively. That is to say, CPCL elements were randomly sampled from the successful parameter configuration list (SPCL). The above process was repeated until the termination condition was satisfied. As a result, the proper parameter set for *PR* and *SF* gradually self-adapted by learning from previous experiences of generating promising solutions. If the SPCL overflows, the earliest records stored in the SPCL were removed so that the current successful parameter sets can be stored in the list, which can avoid any inappropriate long-term accumulation effects. The present study set the length (L) of both CPCL and SPCL at 200. A small variation in length L was verified as not having any significant influence on the performance of the proposed algorithm.

4.4.8 Pareto greedy selection

Considering both a parent solution, $\boldsymbol{X}_i$ and an offspring solution, $\boldsymbol{V}_i$, the better solution is maintained by the Pareto greedy selection in Algorithm 3, where $\boldsymbol{V_i}$ is added into the external archive (*EXA*) only if the new solution, $\boldsymbol{V_i}$, is not dominated by any member of the *EXA*. After such a selection, the number of exploitations, $trial_i$, is incremented by 1, if $\boldsymbol{X}_i$ is maintained ($\boldsymbol{X}_i$ dominates $\boldsymbol{V}_i$). When $trial_i$ exceeds the limit parameter, $\boldsymbol{X}_i$ will be explored by the scouts.

Algorithm 3. Pareto greedy selection ($\boldsymbol{V}_i$, $\boldsymbol{X}_i$)

```
If (V_i dominates X_i)
    X_i = V_i
    trial_i = 0
Else if (V_i and X_i are non-dominated with each other)
    If (V_i is added into EXA)
        X_i = V_i
        trial_i = 0
    Else
        trial_i = trial_i + 1
    End if
Else
    trial_i = trial_i + 1
End if
```

4.4.9 Update external Pareto archive set

As stated in Section 4.4.2, an external archive (*EXA*) was employed to preserve the non-dominated solutions obtained during the search process. At the end of each iteration, the members of the *EXA* were updated. However, the number of Pareto-optimal solutions for most problems is very large and may include an infinite number of individuals. A larger number of Pareto-optimal solutions results in a greater computational burden such that the *EXA* size must be restricted to a predefined value. The present study used the fast non-dominated sorting method to decrease the size of the archive set and maintain the diversity of the Pareto solutions. The crowding distances [50] of all the archive members were calculated and sorted from largest to smallest once the number of non-dominated solutions exceeded the allowed archive size. The top N_{max} (maximum size of the *EXA*) members were maintained, whereas the remaining crowded members were removed to maintain a diverse distribution among the archive members.

4.4.10 The procedure for the proposed hybrid approach

The pseudocode for the proposed HABC is presented in Algorithm 4, where every key element has been explained as before.

5 Experimental studies on benchmark problems

Widely used benchmark problems were used to examine the proposed HABC algorithm in a computational environment with MATLAB R2013b for a 64-bit Windows 7 operating system on a 2.5 GHz PC with 4 GB RAM.

5.1 Test problems

For the performance comparison, the present study selected 21 benchmark problems, that are often used in research to test MOEAs for MOPs, including:

(1) The ZDT bi-objective problems: ZDT1-ZDT4, ZDT6 [53].
(2) The DTLZ tri-objective problems: DTLZ1- DTLZ5, DTLZ7 [54].
(3) The CEC09 benchmark problems: UF1-UF10, wherein UF1-UF7 are bi-objective and UF8-UF10 are tri-objective [55].

5.2 Performance metrics

The performance metrics were classified into three categories depending on their ability to measure convergence, diversity of the obtained solutions, or both criteria. We adopted two widely used metrics, namely the invert generational distance (IGD) [53] and the hyper-volume (HV) [34], to measure both the convergence and diversity of the obtained solutions.

Algorithm 4. The proposed HABC algorithm

```
1:Initialize the food source position solutions (FS). /* see in Section 4.4.3*/
2: FS←Generate employed bees(SN).
3: FS←Fast non-dominated sort(FS, SN).
4: FS←Crowding distance assignment(FS, SN).
5: Add the non-dominated solutions in food sources to the external archive.
6: while <no termination conditions satisfied> do
7:   /* Employed bees Phase in Section 4.4.4 */.
8:   for i=1:SN
9:     Select PR from CPCL;
10:    Update FS_i by employed bee search with multiple dimension perturbation;
11:    PR is added to SPCL and removed from CPCL once FS_i is improved;
12:  End
13:  /* Onlookers Phase in Section 4.4.6*/.
14:  Calculate probabilities p(i) for onlookers by Eq.(25) /* see in Section 4.4.5*/
15:  i←1;s←0;
16:  while s<SN  /*Onlooker bee phase: send onlookers to food sources*/
17:    r←rand(0,1);
18:    if r<p(i) then  /*Stochastic sampling*/
19:      s←s+1; /*Select SF from CPCL*/
20:      Update FS_i by onlooker bee search improved by Levy flight;
21:      SF is Added to SPCL and removed from CPCL once FS_i is improved;
22:    end
23:    i←(i+1) mod (SN);
24:  end
25:  /* Scouts Phase */.
26:  for i=1:SN
27:    If trial_i>limit then  /* Determine the abandoned solutions */
28:      FS_i←randomly generate a new solution by standard scout bee search,
29:      and then set trial_i=0;
30:    End
31:  End
32:  FS←Fast non-dominated sort(FS, SN).
33:  FS←Crowding distance assignment(FS, SN).
34:  Update the external archive/* see in Section 4.4.9*/
35:End while
```

IGD was used to evaluate the average distance between the set of true Pareto solutions, P^*, and the obtained approximation set of non-dominated solutions, P, which can be defined as follows:

$$IGD(P^*, P) = \sqrt{\frac{\sum_{v \in P^*} d(v, P)^2}{|P^*|}} \tag{26}$$

where $d(v, P)$ is the minimum Euclidean distance between v and the points in P, and $|P^*|$is the number of P^* points, wherein the small value of IGD-metric is preferred.

The hyper-volume (HV) criterion is used to assess both the convergence and diversity. Given the approximation set, P, and a reference point, R, HV is transformed into a measurement of the region that is simultaneously dominated by P and bounded above by R, which is formally described as follows:

$$HV(P, R) = Volume(\bigcup_{F \in P} \{x \,|F \prec x \prec R\}) \tag{27}$$

where the reference point, R, for the HV-metric is the worst value in each objective dimension. A higher value of HV signifies a better approximation set.

5.3 Parameter setting

The proposed HABC was compared to several state-of-the-art MOEAs such as NSGA-II [50], AbYSS [56], MOsaDE [57], and SMPSO [58]. The parameter settings are listed below.

- Public parameters:
  - Population size *SN*: 100 for bi-objective problems, 300 for tri-objective problems.
  - Maximal function evaluations FEs: 300,000.
- Parameters of NSGA-II are set the same as those in [50]:
  - Crossover probability: 0.9
  - SBX distribution index: 20
  - Mutation probability: 1/*n*
  - Mutation distribution index: 20
- Parameters of AbYSS are set the same as those in [56]:
  - Distribution index of the polynomial mutation: 20
  - Size of reference set: 10
  - Crossover probability: 1
  - Mutation probability: 1/*n*
- Parameters of MOsaDE are set the same as those in [57]:
  - Crossover probability *CR*: with normal distribution of mean 0.5 and standard deviation 0.1.
  - Scaling factor *SF*: with linearly reducing mean value from 1.0 to 0.05 and standard deviation 0.1
- Parameters of SMPSO are set the same as those in [58]:
  - Mutation distribution index: 20
  - Mutation probability: 1/*n*

where n is the number of decision variables. In addition, 30 runs were independently conducted for each algorithm with respect to each instance to obtain statistically sound conclusions. Likewise, the parameter limit for HABC was set at 100.

5.4 Experimental results

Tables 1 and 2 respectively present the mean and standard deviation of the HV and IGD metric values of the non-dominated solutions obtained by each algorithm for the 21 problems. The paired Wilcoxon signed-rank test was conducted to examine the statistical significance between HABC and the other algorithms. In Tables 1 and 2, signs "+", "=", and "−" indicate a respectively better than, similar to, or worse HABC performance as compared to its competitor based on the Wilcoxon signed-ranked test at a 0.05 significance level. The results are summarized as "$w/t/l$" on the last row of the two tables, which counts the number of problems that the proposed HABC significantly outperforms, performs similar to, or worse than its competitor in the corresponding column. In addition, the best results are highlighted in boldface based on the average metric value.

The proposed HABC algorithm provided the best results for 16 out of the 21 problems based on the average metric values in Table 1, wherein MOsaDE had the best

Table 1 Comparison results of HV between HABC and other MOEAs

HV	HABC	NSGA-II	AbYSS	MOsaDE	SMPSO
ZDT1	**6.656e−01(1.741e−05)**	6.602e−01(2.648e−04)+	6.621e−01(1.601e−05)+	6.619e−01(2.509e−05)+	6.621e−01(1.370e−05)+
ZDT2	**3.323e−01(1.991e−05)**	3.272e−01(3.203e−04)+	3.288e−01(1.745e−05)+	3.286e−01(2.795e−05)+	3.288e−01(1.922e−05)+
ZDT3	**5.167e−01(4.075e−06)**	5.153e−01(1.004e−04)+	5.159e−01(1.055e−05)+	5.158e−01(1.361e−05)+	5.159e−01(7.500e−06)+
ZDT4	**6.656e−01(4.136e−06)**	6.608e−01(2.410e−04)+	6.620e−01(7.093e−05)+	6.618e−01(2.353e−05)+	6.620e−01(1.868e−05)+
ZDT6	**4.054e−01(3.256e−07)**	3.981e−01(5.567e−04)+	4.006e−01(7.210e−05)+	4.013e−01(1.841e−05)+	4.014e−01(4.975e−05)+
DTLZ1	**8.001e−01(2.030e−04)**	7.619e−01(6.769e−03)+	7.612e−01(7.126e−03)+	7.679e−01(4.621e−03)+	7.413e−01(5.163e−03)+
DTLZ2	**4.447e−01(6.227e−04)**	3.867e−01(4.595e−03)+	3.952e−01(5.434e−03)+	3.924e−01(3.405e−03)+	3.737e−01(5.542e−03)+
DTLZ3	**4.410e−01(8.073e−04)**	3.877e−01(5.471e−03)+	3.897e−01(4.642e−03)+	3.861e−01(3.519e−03)+	3.720e−01(3.898e−03)+
DTLZ4	**4.506e−01(1.952e−02)**	3.997e−01(4.244e−03)+	4.092e−01(4.363e−03)+	3.988e−01(4.192e−03)+	3.858e−01(6.098e−03)+
DTLZ5	**9.483e−02(5.494e−06)**	9.234e−02(1.626e−04)+	9.354e−02(1.721e−05)+	9.350e−02(2.316e−05)+	9.354e−02(2.235e−05)+
DTLZ7	3.083e−01(1.181e−02)	2.993e−01(3.584e−03)+	2.651e−01(3.284e−02)+	**3.086e−01(3.003e−03)**−	2.916e−01(4.577e−03)+
UF1	**6.655e−01(5.364e−05)**	5.596e−01(3.899e−02)+	5.512e−01(4.518e−02)+	5.700e−01(1.281e−02)+	5.626e−01(9.477e−03)+
UF2	**6.631e−01(1.035e−03)**	6.310e−01(7.548e−03)+	6.363e−01(6.228e−03)+	6.330e−01(6.879e−03)+	6.353e−01(3.097e−03)+
UF3	**6.584e−01(9.183e−03)**	4.823e−01(3.990e−02)+	4.021e−01(6.646e−02)+	3.149e−01(3.798e−02)+	4.822e−01(5.702e−02)+
UF4	3.391e−01(5.298e−03)	3.484e−01(1.254e−03)−	3.439e−01(4.384e−03)−	**3.600e−01(3.475e−04)**−	3.452e−01(3.176e−03)−
UF5	7.682e−02(1.000e−01)	**1.734e−01(7.114e−02)**−	1.663e−01(5.257e−02)−	0.000e+00(0.000e+00)+	0.000e+00(0.000e+00)+
UF6	2.192e−01(8.181e−02)	2.435e−01(6.736e−02)=	**2.590e−01(5.482e−02)**=	1.336e−01(1.208e−01)+	2.057e−02(8.995e−03)+
UF7	**4.990e−01(1.724e−04)**	3.880e−01(9.360e−02)+	3.061e−01(1.035e−01)+	4.528e−01(3.492e−03)+	4.595e−01(4.837e−03)+
UF8	**7.067e−01(8.530e−03)**	4.192e−01(2.539e−01)+	4.135e−01(1.823e−01)+	1.102e−01(2.102e−02)+	2.003e−01(1.237e−01)+
UF9	**9.276e−01(6.296e−02)**	7.678e−01(1.360e−01)+	7.402e−01(1.305e−01)+	8.801e−01(1.098e−01)+	4.499e−01(1.147e−01)+
UF10	1.088e−01(2.608e−02)	5.778e−02(5.583e−02)+	1.272e−01(1.018e−01)=	8.896e−02(5.649e−02)=	**1.728e−01(1.052e−02)**−
$w/t/l$		'18/1/2'	'17/2/2'	'18/1/2'	'19/0/2'

Table 2 Comparison results of IGD between HABC and other MOEAs

IGD	HABC	NSGA-II	AbYSS	MOsaDE	SMPSO
ZDT1	**1.567e−05(1.280e−07)**	5.532e−05(2.609e−06)+	3.873e−05(4.160e−07)+	4.127e−05(4.272e−07)+	3.875e−05(4.630e−07)+
ZDT2	**7.325e−06(9.781e−09)**	5.345e−05(2.539e−06)+	3.807e−05(3.358e−07)+	4.072e−05(3.890e−07)+	3.801e−05(3.639e−07)+
ZDT3	**1.746e−05(2.599e−08)**	3.827e−05(1.478e−06)+	2.718e−05(5.833e−07)+	2.849e−05(4.342e−07)+	2.707e−05(4.667e−07)+
ZDT4	**1.562e−05(3.504e−08)**	5.267e−05(2.210e−06)+	3.849e−05(3.432e−07)+	4.097e−05(4.011e−07)+	3.847e−05(3.095e−07)+
ZDT6	**3.287e−06(3.797e−10)**	4.955e−05(3.619e−06)+	2.895e−05(1.162e−07)+	3.039e−05(2.478e−07)+	2.903e−05(5.168e−07)+
DTLZ1	**1.344e−04(2.262e−07)**	4.616e−04(3.864e−05)+	4.659e−04(4.751e−05)+	3.964e−04(1.473e−05)+	4.584e−04(1.651e−05)+
DTLZ2	**1.887e−04(7.805e−07)**	6.368e−04(2.599e−05)+	6.461e−04(3.424e−05)+	5.838e−04(1.753e−05)+	6.268e−04(2.302e−05)+
DTLZ3	**1.905e−04(8.446e−07)**	6.219e−04(3.132e−05)+	6.387e−04(3.653e−05)+	5.621e−04(1.221e−05)+	6.043e−04(2.784e−05)+
DTLZ4	**3.001e−04(4.868e−04)**	6.443e−04(2.647e−05)+	6.077e−04(2.253e−05)+	6.013e−04(1.733e−05)+	6.521e−04(2.309e−05)+
DTLZ5	**2.458e−05(4.699e−08)**	7.770e−05(3.120e−06)+	5.404e−05(7.576e−07)+	5.605e−05(7.191e−07)+	5.362e−05(7.471e−07)+
DTLZ7	6.774e−04(1.209e−03)	4.408e−04(5.321e−05)−	3.835e−03(2.124e−03)+	**4.235e−04(3.043e−05)**−	4.735e−04(3.467e−05)−
UF1	**1.368e−05(7.649e−07)**	1.263e−03(4.644e−04)+	1.411e−03(5.554e−04)+	1.057e−03(1.412e−04)+	7.810e−04(6.678e−05)+
UF2	**3.535e−05(8.159e−06)**	3.267e−04(1.049e−04)+	3.097e−04(1.372e−04)+	2.448e−04(6.255e−05)+	2.080e−04(2.380e−05)+
UF3	**7.253e−05(1.144e−04)**	6.936e−03(1.579e−03)+	8.021e−03(2.420e−03)+	2.379e−03(3.507e−04)+	1.327e−03(4.489e−04)+
UF4	8.273e−04(1.532e−04)	5.013e−04(2.673e−05)−	7.217e−04(1.871e−04)−	**2.459e−04(5.106e−06)**−	6.228e−04(1.089e−04)−
UF5	2.901e−03(2.754e−03)	5.872e−03(2.980e−03)+	6.038e−03(3.237e−03)+	**2.518e−03(1.312e−06)**−	2.300e−02(8.958e−03)+
UF6	**2.693e−03(2.626e−03)**	4.572e−03(2.338e−03)+	5.892e−03(1.573e−03)+	3.816e−03(1.577e−03)+	7.539e−03(1.277e−03)+
UF7	**1.459e−05(3.093e−06)**	2.184e−03(2.359e−03)+	4.185e−03(2.508e−03)+	6.351e−04(4.774e−05)+	3.535e−04(6.032e−05)+
UF8	**5.130e−04(2.050e−04)**	3.745e−03(2.243e−03)+	3.573e−03(1.754e−03)+	6.496e−03(1.772e−04)+	5.534e−03(1.392e−03)+
UF9	**6.161e−04(6.008e−04)**	2.518e−03(9.748e−04)+	2.646e−03(8.526e−04)+	1.867e−03(9.319e−04)+	5.618e−03(1.125e−03)+
UF10	8.304e−03(1.614e−03)	7.768e−03(2.472e−03)−	7.990e−03(2.593e−03)=	6.020e−03(2.013e−03)−	**4.321e−03(4.544e−04)**−
$w/t/l$		'18/0/3'	'19/1/1'	'17/0/4'	'18/0/3'

performance for two problems, SMPSO exhibited the best performance on UF10, NSGA-II performed the best on UF5, and AbYSS offered the best solution for UF6. In addition, HABC achieved an overwhelming advantage over the other competitors because HABC outperformed NSGA-II 18 problems, AbYSS 17 problems, MOsaDE 18 problems, and SMPSO 19 problems according to pairwise comparison results summarized on the last line of Table 1.

More detailed information can be obtained if the results were displayed using boxplots, which depict the distribution of the numerical data. Figure 4 shows the HV value distributions for each algorithm's final solutions on the tested problems, from which HABC exhibited a significantly higher and narrower boxplot as compared to those of other rivals for all the problems except for DTLZ7, UF4-UF6, and UF10. Given that HV is the performance metric that illustrates both the diversity and convergence of an algorithm, it can be stated that the Pareto optimal solutions obtained by HABC are close to and highly distributed along the true Pareto front.

A careful inspection of Fig. 4 reveals a slightly similar algorithm performance ranking order for certain types of problems. For ZDT bi-objective problems, HABC ranked first place among the contestant algorithms, NSGA-II was outperformed by all the other contestant algorithms and ranked last. Other algorithms such as AbYSS, MOsaDE, and SMPSO obtained similar results. However, SMPSO was considered the worst algorithm instead of NSGA-II for DTLZ problems, except for DTLZ5 and DTLZ7. HABC still exhibited the best performance on all DTLZ problems. AbYSS, MOsaDE, and SMPSO obtained similar results. The algorithms perform differently for different problems given that CEC09 problems contained composite types of problems.

The IGD metric results of the compared algorithms in Table 2 depict HABC yielding the best results for 17 of the 21 problems, whereas SMPSO and MOsaDE achieved only one and three best results, respectively, and NSGA-II and AbYSS failed to obtain any. The Wilcoxon signed-rank test result presented in the last row of Table 2 also illustrates a better HABC performance for the majority of the 21 problems by respectively obtaining 18, 19, 17, and 18 significantly better results as compared to its corresponding competitors. Given that IGD is a performance metric used to measure the convergence and diversity of the obtained solutions using an algorithm, the aforementioned results indicate that the obtained approximate Pareto fronts using HABC are more diversified and closer to the optimal Pareto fronts than those from the other algorithms.

The IGD value distributions obtained from the compared algorithms in Fig. 5 depict a narrower HABC boxplot as compared to those of its peers except for UF4, UF6, and UF10, which also shows stability of HABC in converging

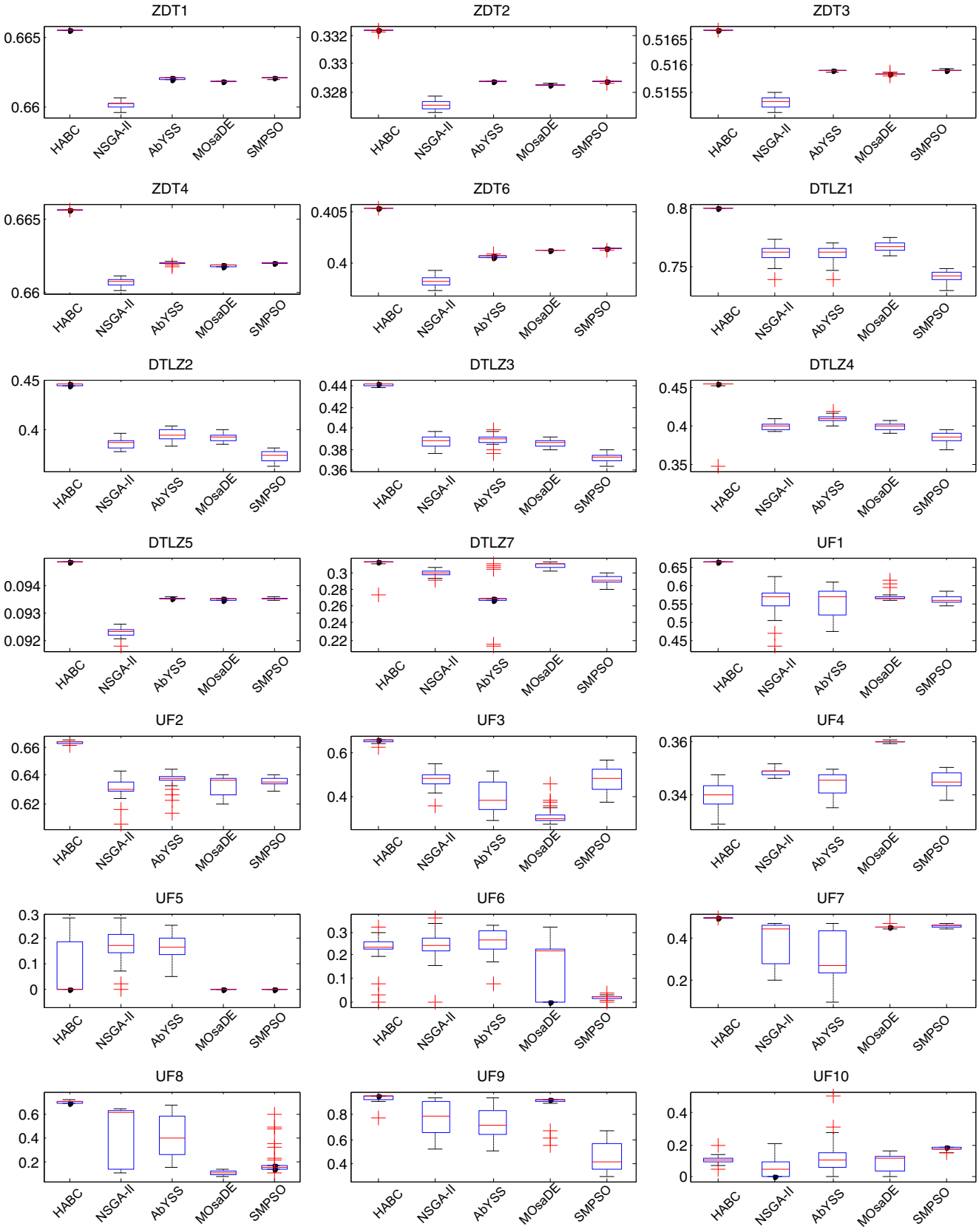

Fig. 4 Boxplots for the HVs obtained using the compared algorithms

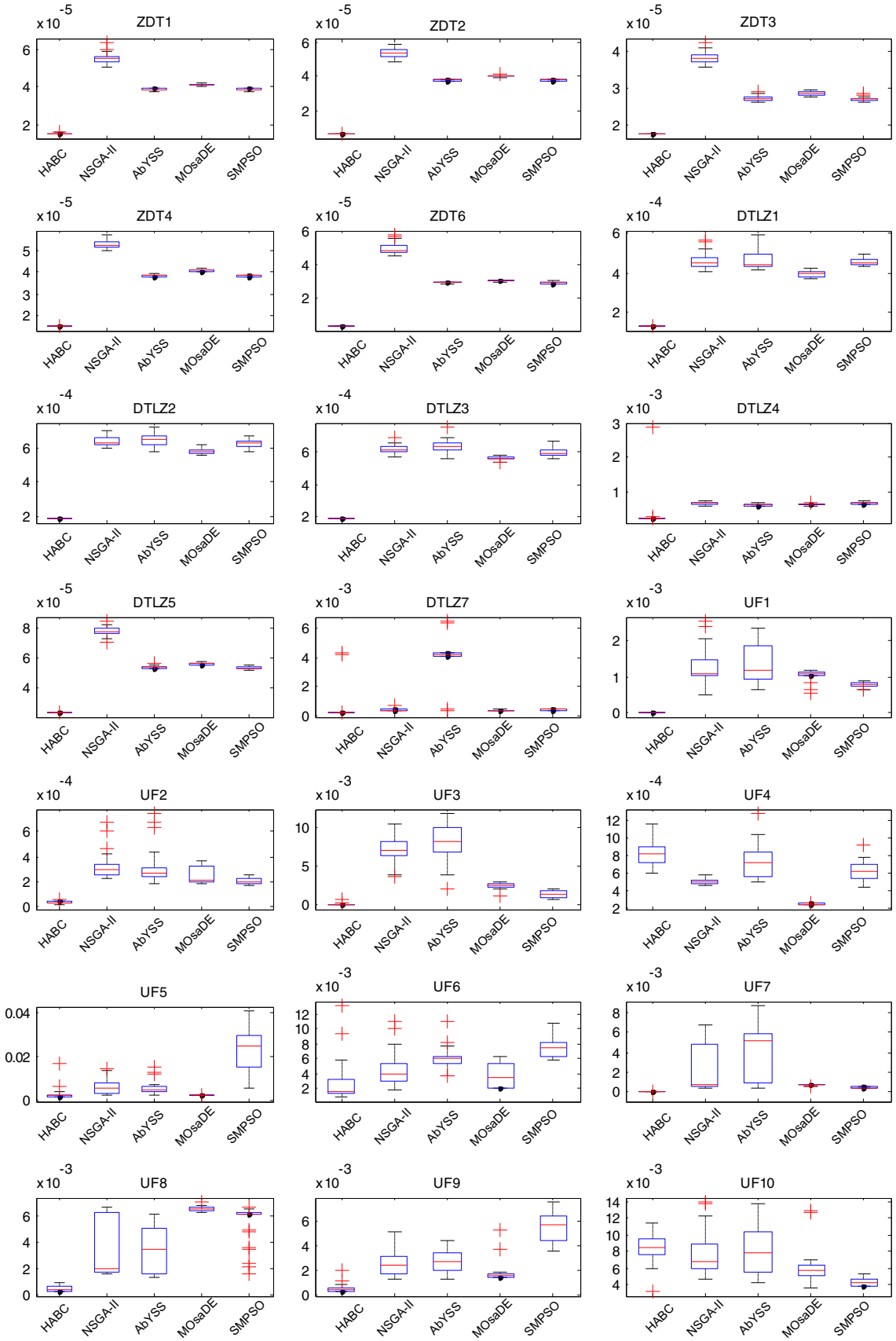

Fig. 5 Boxplots of the IGD obtained from the compared algorithms

towards the true Pareto front. However, it should be pointed out that although HABC performed better than the other algorithms for a majority of the problems, it did not work well on several special problems such as DTLZ7, UF4, UF5, and UF10, wherein HABC was outperformed by its competitors in terms of both IGD and HV metrics. MOsaDE performed well in solving DTLZ7 and UF4, while SMPSO worked well for UF10.

These results can be explained by the No Free Lunch Theorem (NFLT) [59]. According to the NFLT, an algorithm cannot dominate another algorithm for all the problems and all the aspects, as a general-purpose universal optimization algorithm is theoretically impossible. Therefore, no singular strategy can be expected to outperform another for all types of problems.

The distributions of the final solutions with the median IGD values obtained by the algorithms with respect to UF1,UF3, and UF7 are presented in Figs. 6–8, respectively, to provide a graphical overview of the behavior of these algorithms. Figure 6 illustrates the ability of HABC in covering the entirety of the whole Pareto front. The Pareto fronts obtained by NSGA-II and MOsaDE were slightly similar and missed a portion of the central part of the true Pareto front. However, the Pareto front from AbYSS missed both the left and right sides of the true Pareto front. The solutions obtained by SMPSO had a farther distance to the true Pareto front despite having good diversity.

HABC still had the best convergence on UF3 based on the results presented in Fig. 7. The solutions obtained by NSGA-II and AbYSS were only partly close to the true Pareto front, whereas MOsaDE found solutions farther from the true Pareto front and SMPSO had solutions at the ends of the Pareto front.

The superiority of HABC is also observed in Fig. 8. However, the solutions obtained by the SMPSO algorithm were better than those of AbYSS and MOsaDE, though all of them missed a part of the true Pareto front. The worst results belonged to NSGA-II, which only obtained a small discontinuous part of solutions on the right side of the true Pareto front.

According to the above-presented comparison and statistical analysis, the proposed HABC performed the best among the five algorithms in terms of the convergence rate, stability, and solutions' diversity. Fast HABC convergence originated from the solution-updating mechanism of the employed bee phase, where multiple dimensions of the solutions were perturbed at each iteration. Following the exploration of the employed bee phase, onlooker bees tended to exploit the promising regions of the search space provided by Lévy flight with adaptive step length. In addition, occasional large-steps or long-distance jumps are allowed in the Lévy distribution, which reemphasizes exploration and improves the diversity of the solutions across the Pareto front.

5.5 Effectiveness of improvement strategies

5.5.1 Impacts of parameter settings

PR and *SF* are two parameters used in HABC that control the convergence speed and adjust the exploration and exploitation. The perturbation rate, *PR*, was employed to adjust the parameter variation frequency when a neighboring solution is produced. Likewise, the scaling factor, *SF*, was used to determine the step length of the Lévy flight in the onlooker bee phase. To investigate the impacts of *PR* and *SF* on the performance of HABC, we tested 36 combinations of 9 *PR*

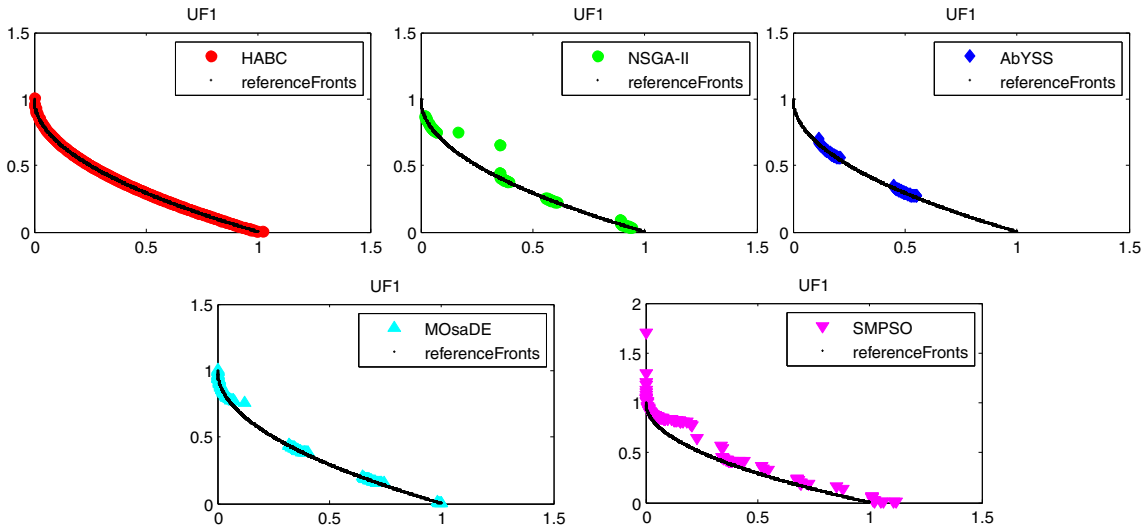

Fig. 6 Distribution for the solutions obtained using the algorithms on UF1

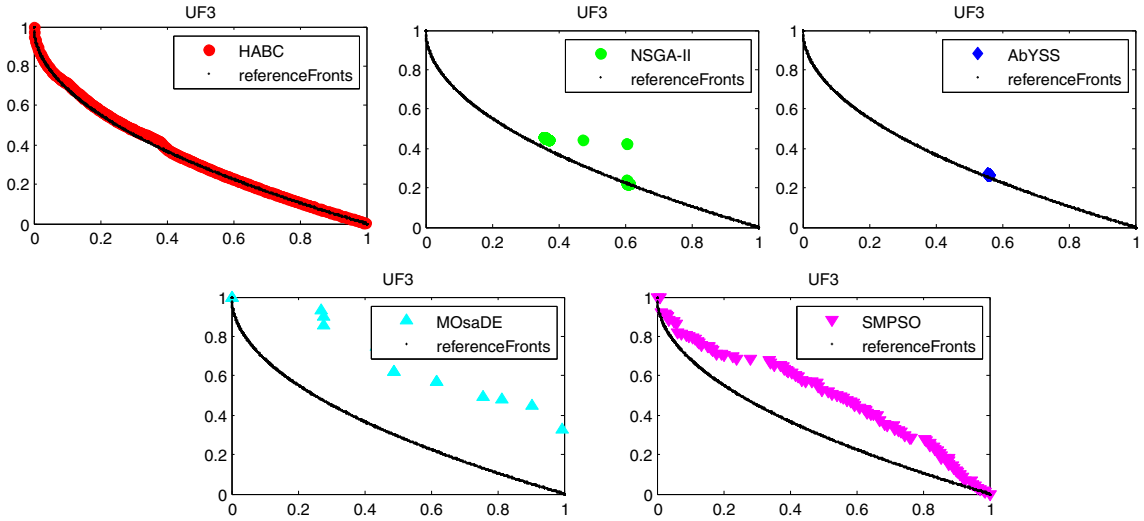

Fig. 7 Distribution for the solutions obtained using the algorithms on UF3

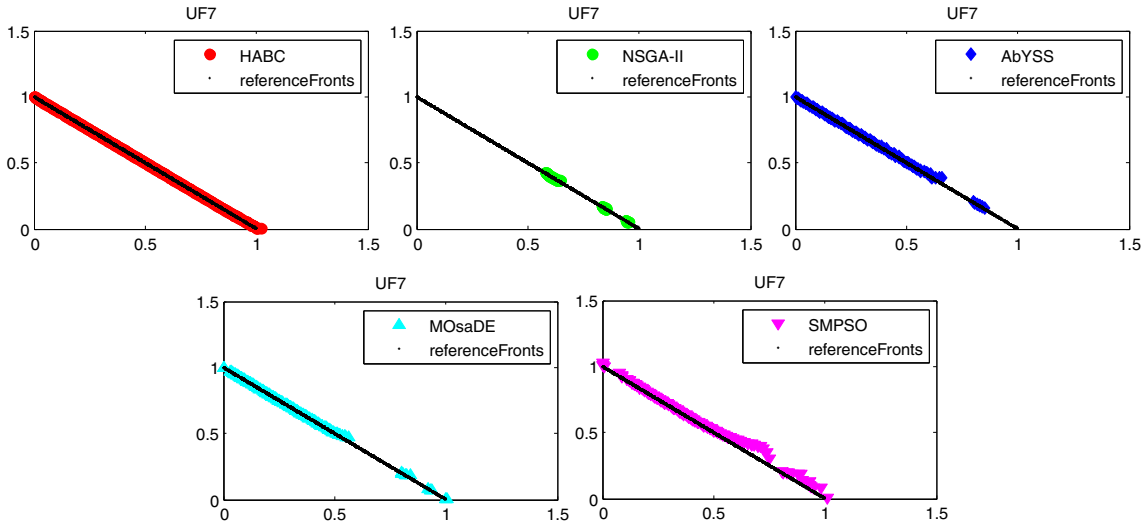

Fig. 8 Distribution for the solutions obtained using the algorithms on UF7

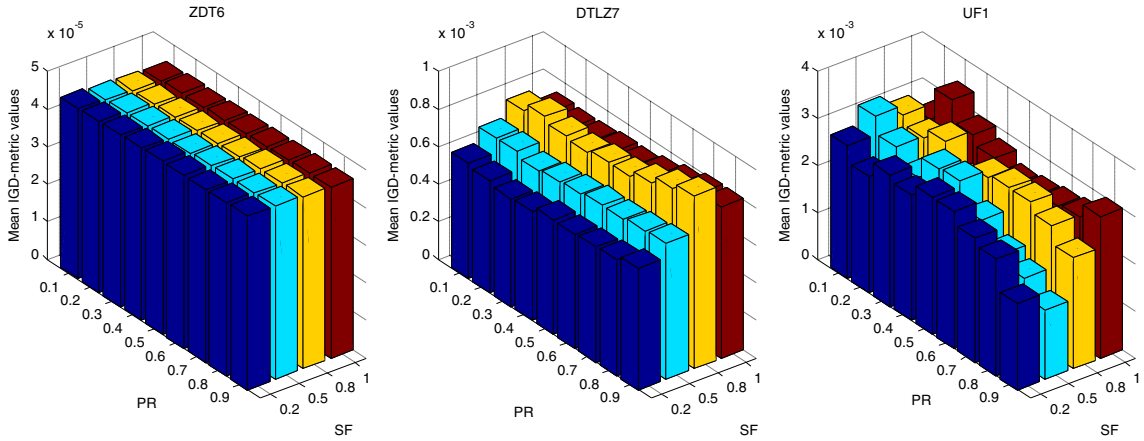

Fig. 9 Mean IGD-metric values obtained by HABC with different parameter combinations of PR and SF on ZDT6, DTLZ7, and UF1 problems

values (i.e., from 0.1 to 0.9 with a step size of 0.1) and 4 *SF* values (0.2, 0.5, 0.8, 1) on ZDT6, DTLZ7, and UF1 problems, which were chosen to represent the ZDT, DTLZ, and UF problems, respectively. The other parameters remained the same as in Section 5.3. Thirty independent runs were conducted for each combination of *PR* and *SF* on each of the tested problems.

The mean IGD-metric values obtained by HABC with all the combinations of *PR* and *SF* are presented in Fig. 9, wherein the settings of *PR* and *SF* had the least effect on ZDT6, a significant effect on DTLZ7, and the most effect on UF1 given that the search space of UF1 is more difficult than those of ZDT6 and DTLZ7. The performance of HABC in solving a specific problem crucially depends on appropriate parameter settings, and employing a trial-and-error scheme to search for the most suitable combination of *PR* and *SF* is time-consuming. Therefore, the adaptive *PR* and *SF* settings may be critical for improving the performance of HABC for some problems.

5.5.2 Effectiveness of the adaptive perturbation rate

To verify the effect of the adaptive perturbation rate, HABC (with an adaptive *PR* as described above) was compared with its four variants at fixed perturbation rates, each of which only adopted a fixed *PR* (0.1, 0.3, 0.5, or 0.7) for the entire evolution process. The Friedman test [60] was used to determine the accumulation of wins for each algorithm by a non-parametric multiple comparisons test, the computational results of which are presented in Fig. 10, where the horizontal axis denotes the parameter setting of *PR* and the vertical axis represents the accumulation of wins for HABC and the variants (denoted as 0.1, 0.3, 0.5, and 0.7 for simplicity) when solving ZDT, DTLZ, and UF problems, respectively.

The best fixed value of *PR* in Fig. 10 differs for different types of problems given that a value of 0.7 resulted in the second best result for ZDT problems, while the same value gave the worst result for DTLZ problems. Likewise, the results were unsuccessful in determining a fixed *PR* setting that always provided the best solution for all the problems. In addition, the difference between the best and worst algorithms was more evident with increasing problem complexity. Nevertheless, HABC with adaptive *PR* was better than its variants at all times. Therefore, the adaptive perturbation rate was concluded to enhance the effectiveness and robustness of the proposed HABC when solving different problems.

5.5.3 Effectiveness of the adaptive scaling factor

Similarly, the effect of the adaptive scaling factor was tested by comparing the proposed HABC with its four variants at fixed scaling factors (denoted as 0.2, 0.5, 0.8, and 1 for simplicity) during the evolution processes. The comparison results are presented in Fig. 11, where the accumulation of the wins for each variant is presented in the vertical axis, and the horizontal axis denotes a comparison between the algorithms. None of the four variants with fixed scaling factors obtained the best result for all of the problems, which is similar to previous observations presented in Section 5.5.2. In general, the adaptive scaling factor can be concluded as positive for the proposed HABC algorithm.

6 Case study on multi-objective SCOS problems in CMfg

In this section, the proposed algorithm was applied to engineering optimization problems, particularly SCOS problems in CMfg. CMfg resources include both software applications and various types of manufacturing equipment. One typical manufacturing project adapted from [7] was extended as a case study. A total set of 30 tasks in the cloud manufacturing environment was presented as a directed acyclic graph (DAG) (Fig. 12), in which each node represents a task unit, the color of the node represents its task type, each directed line between two nodes represents the communication relationship among the tasks, all tasks strictly observe the tasks' priority rules, and a successor task node can only be started following the completion of all predecessor tasks.

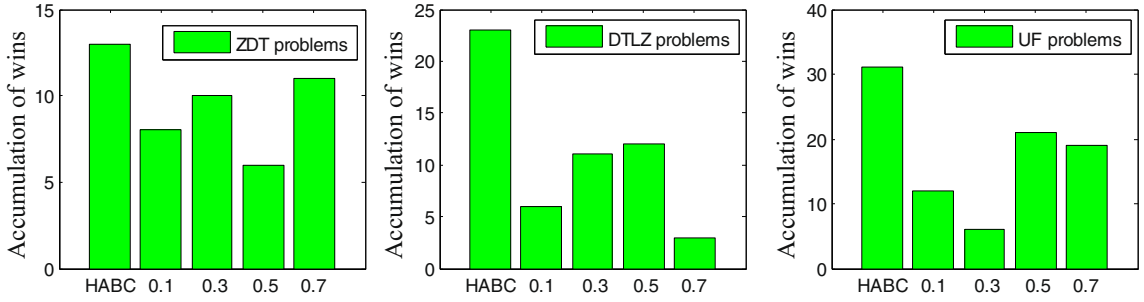

Fig. 10 Accumulation wins for HABC and its variants at a fixed perturbation rate based on the Wilcoxon rank-sum test of the IGD metric

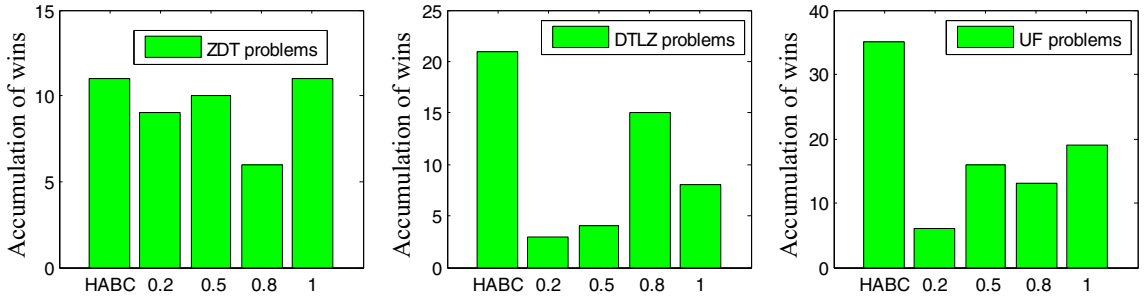

Fig. 11 Accumulation wins for HABC and its variants at fixed scaling factors based on the Wilcoxon rank-sum test of the IGD metric

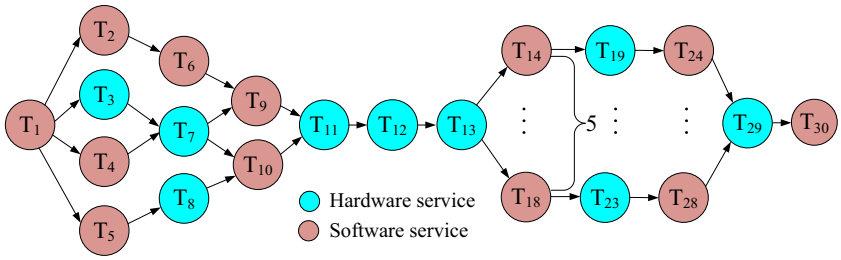

Fig. 12 DAG for the manufacturing project

Table 3 Parameter ranges involved in QoS and energy consumption for the candidate services

QoS Parameter				*T*			*Av*		*Re*		*Re*
Range				[1,10]			[0,1]		[0,1]		[0,1]
En Parameter	*s*	*v*	*p*	*c*	*q*	*g*	ζ	η	*e*	*f*	*u*
Range	[1,10]	[1,10]	[1,5]	10	[1,10]	[10,50]	[1,10]	[1,10]	[10,100]	0	0

Table 4 Comparison results of HV for the five algorithms presented in the studied case

HV	HABC	NSGA-II	AbYSS	MOsaDE	SMPSO
30_20	**4.537e+01(9.078e-01)**	4.323e+01(1.432e+00)+	3.252e+01(1.460e+00)+	3.659e+01(9.480e+00)+	1.522e+01(6.481e+00)+
30_50	**1.420e+02(2.239e+00)**	1.340e+02(8.682e+00)+	1.054e+02(6.903e+00)+	6.362e+01(1.410e+01)+	6.253e+01(6.921e+00)+
30_80	**7.300e+01(6.350e-01)**	6.877e+01(2.302e+00)+	4.250e+01(4.620e+00)+	1.455e+01(1.130e+01)+	1.060e+01(7.882e+00)+
30_100	**8.500e+01(1.700e+00)**	7.886e+01(3.593e+00)+	5.383e+01(4.847e+00)+	1.881e+01(1.265e+01)+	1.514e+01(5.031e+00)+
$w/t/l$		'4/0/0'	'4/0/0'	'4/0/0'	'4/0/0'

Table 5 Comparison results of IGD for the five algorithms presented in the studied case

IGD	HABC	NSGA-II	AbYSS	MOsaDE	SMPSO
30_20	**1.079e+00(2.444e-01)**	2.415e+00(1.570e+00)+	3.442e+00(1.009e+00)+	1.133e+00(3.114e-01)=	8.509e+00(1.729e+00)+
30_50	**2.627e+00(1.273e+00)**	5.957e+00(2.906e+00)+	5.550e+00(1.931e+00)+	1.113e+01(1.845e+00)+	9.861e+00(1.431e+00)+
30_80	**9.741e-01(1.971e-01)**	3.782e+00(7.189e-01)+	3.908e+00(1.848e+00)+	1.710e+01(4.234e+00)+	1.541e+01(2.176e+00)+
30_100	**1.290e+00(7.996e-01)**	4.383e+00(1.737e+00)+	4.808e+00(1.198e+00)+	1.712e+01(5.226e+00)+	1.345e+01(2.585e+00)+
$w/t/l$		'4/0/0'	'4/0/0'	'3/1/0'	'4/0/0'

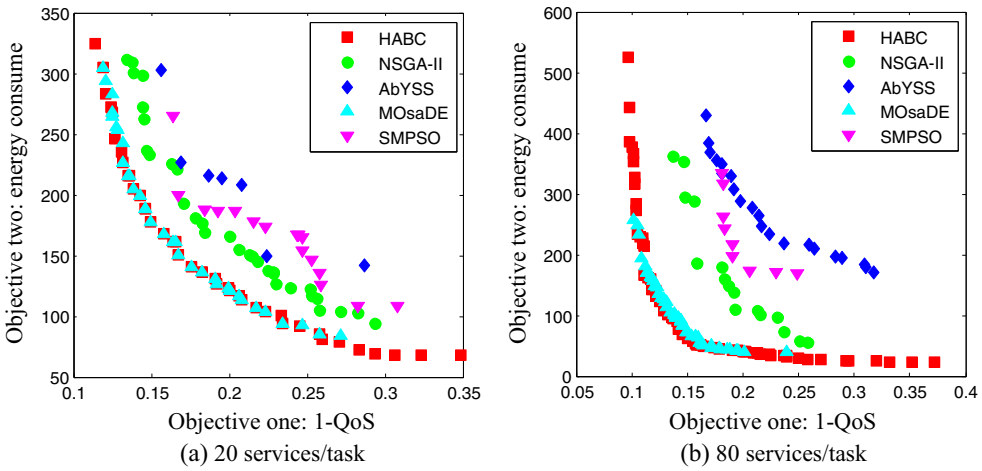

Fig. 13 Pareto solutions obtained by the algorithms: (a) 20 candidate services per task; (b) 80 candidate services per task

To study the scalability of HABC in solving SCOS problems, we set the available number of cloud services for each task at 20, 50, 80, and 100, respectively. The value ranges of parameters involved in both QoS and energy consumption for candidate manufacturing services (MSs) are set as presented in Table 3. All parameter values involved in the candidate services were randomly generated within the specified ranges and stored in a *.txt* file for consistent initial conditions.

NSGA-II, AbYSS, MOsaDE, and SMPSO were used as competitors to address the multi-scaled SCOS problems when investigating the performance of HABC. The experimental settings were identical to those mentioned in Section 5.3, with 30 experimental runs conducted for each tested problem. Note that the true Pareto front is unknown in the SCOS problems. Therefore, in the IGD calculation, true Pareto front was assumed as the combined non-dominated Pareto front solutions that were obtained from all of the considered algorithms for a large number of cycles. The IGD values of the true Pareto front to the non-dominant solutions from the respective algorithms were then calculated. Tables 4 and 5 present the mean and standard deviation of the HV-metric and IGD-metric values for the solutions obtained by the compared algorithms on the 4 scales presented in the studied case.

In addition, the Wilcoxon rank-sum test set at a 5 % significance level was performed to compare the significance of the differences between the compared algorithms. Based on the IGD comparison in Table 5, HABC exhibited a better performance than NSGA-II, AbYSS, MOsaDE, and SMPSO as it resulted in 4, 4, 3, and 4 wins out of 4, respectively. The "+", "= ", and "–" signs in the table also maintained its meaning. Likewise, '$w/t/l$' on the last row of the table denotes the number of HABC wins (w), ties (t), and loses (l) as compared to its corresponding competitor.

The above comparisons clearly illustrate a significantly better HABC performance for the studied case in terms of both HV and IGD metrics, which validates the advantages of the HABC in solving for SCOS problems.

Algorithm comparisons based on the studied SCOS at two different scales for the candidate services are presented in Fig. 13. The graph on the left side of the figure represents the obtained non-dominant solutions from the case that presented 20 candidate services per subtask, while the graph on the right side presents a case having 80 candidate services per subtask. HABC is observed to be more significantly advantageous in tackling problems with more services. This is due to that the adaptive parameter adjustment strategy is especially effective in enhancing the search capability of the proposed algorithm when solving complicated problems. From the above comparisons, the proposed HABC algorithm appears to be a very promising approach and can be used an alternative tool for solving multi-objective SCOS problems in CMfg.

7 Conclusions

Although evolutionary algorithms have been widely investigated in handling various SCOS problems, they are most focused on single-objective optimization and rarely consider multi-objective SCOS problems in CMfg. This paper proposed HABC, which is a novel multi-objective optimization approach that can be utilized in addressing such multi-objective SCOS problems. In the proposed approach, the multi-dimension permutation mechanism was introduced to the employed bee phase to determine the number of parameters for a given solution that is to be perturbed. Moreover, the cuckoo search-inspired Lévy flight was employed in the onlooker bee phase, which formed a random walk process that obeyed a power-law step-length distribution with a heavy tail. This mechanism allowed more efficient search space explorations on the global scale as to avoid being trapped in the local optima. Furthermore, the perturbation

rate of the employed bee search and the step length of the Lévy flight in the onlooker bee phase was adaptively adjusted by learning its search history. To validate the effectiveness of the proposed approach, a series of MOPs and a practical case was chosen. Its comparison results revealed a better HABC algorithm performance in terms of the solution quality.

Given the superior performance of HABC, It would be interesting to work with other multi-objective approaches based on the swarm intelligence applied in the service composition of CMfg. Since large-scale SCOS problems may require significant CPU time to find a Pareto solution set, the combined application of parallel computing and evolutionary algorithms appears to be promising. Moreover, the proposed algorithm can be further analyzed to develop more enhanced methods of solving other MOPs.

Acknowledgments The project was supported by the National Natural Science Foundation of China under grant Nos. 51675186 and 51175187, the Science & Technology Foundation of Guangdong Province under grant No. 2016A020228005, and the Science & Technology Program of Zhanjiang City under grant No. 2015A01001. The authors would like to thank the Editors and the anonymous referees for their valuable comments and suggestions.

References

1. Li BH, Zhang L, Wang SL, Tao F, Cao JW, Jiang XD, Song X, Chai XD (2010) Cloud manufacturing: a new service-oriented networked manufacturing model. Comput Integr Manuf Syst 16(1):1–16
2. Tao F, Cheng Y, Xu LD, Zhang L, Li BH (2014) CCIOt-CMfg: Cloud Computing and Internet of Things-Based Cloud Manufacturing Service System. IEEE Trans Ind Inf 10(2):1435–1442
3. Tianri W, Shunsheng G, Chi-Guhn L (2014) Manufacturing task semantic modeling and description in cloud manufacturing system. Int J Adv Manuf Technol 71(9-12):2017–2031
4. Luo Y, Zhang L, Tao F, Ren L, Liu Y, Zhang Z (2013) A modeling and description method of multidimensional information for manufacturing capability in cloud manufacturing system. Int J Adv Manuf Technol 69(5-8):961–975
5. Liu N, Li X, Shen W (2014) Multi-granularity resource virtualization and sharing strategies in cloud manufacturing. J Netw Comput Appl 46:72–82
6. Tao F, Zuo Y, Xu LD, Zhang L (2014) Iot-based Intelligent Perception and Access of Manufacturing Resource Toward Cloud Manufacturing. IEEE Trans Ind Inf 10(2):1547–1557
7. Tao F, LaiLi Y, Xu L, Zhang L (2013) FC-PACO-RM: A parallel method for service composition Optimal-Selection in cloud manufacturing system. IEEE Trans Ind Inf 9(4):2023–2033
8. Huang B, Li C, Tao F (2014) A chaos control optimal algorithm for QoS-based service composition selection in cloud manufacturing system. Enterp Inf Syst 8(4):445–463
9. Laili Y, Tao F, Zhang L, Cheng Y, Luo Y, Sarker BR (2013) A Ranking Chaos Algorithm for dual scheduling of cloud service and computing resource in private cloud. Comput Ind 64(4):448–463
10. Seghir F, Khababa A (2016) A hybrid approach using genetic and fruit fly optimization algorithms for QoS-aware cloud service composition. J Intell Manuf. doi:10.1007/s10845-10016-11215-10840
11. Wang D, Yang Y, Mi Z (2015) A genetic-based approach to web service composition in geo-distributed cloud environment. Comput Electr Eng 43:129–141
12. Wang Z, Liu Z, Zhou X, Lou Y (2011) An approach for composite web service selection based on DGQos. Int J Adv Manuf Technol 56(9-12):1167–1179
13. Huo Y, Zhuang Y, Gu J, Ni S, Xue Y (2015) Discrete gbest-guided artificial bee colony algorithm for cloud service composition. Appl Intell 42(4):661–678
14. Zhang H, Zhu BC, Li YP, Yaman O, Roy U (2015) Development and utilization of a Process-oriented Information Model for sustainable manufacturing. J Manuf Syst 37:459–466
15. Dubey R, Gunasekaran A, Childe SJ, Wamba SF, Papadopoulos T (2016) The impact of big data on world-class sustainable manufacturing. Int J Adv Manuf Technol 84(1-4):631–645
16. Wang Z, Subramanian N, Gunasekaran A, Abdulrahman MD, Liu C (2015) Composite sustainable manufacturing practice and performance framework: Chinese auto-parts suppliers' perspective. Int J Prod Econ 170:219–233
17. Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Glob Optim 39(3):459–471
18. Han YY, Liang JJ, Pan QK, Li JQ, Sang HY, Cao NN (2013) Effective hybrid discrete artificial bee colony algorithms for the total flowtime minimization in the blocking flowshop problem. Int J Adv Manuf Technol 67(1-4):397–414
19. Chaves-Gonzalez JM, Vega-Rodriguez MA, Granado-Criado JM (2013) A multiobjective swarm intelligence approach based on artificial bee colony for reliable DNA sequence design. Eng Appl Artif Intel 26(9):2045–2057
20. Metlicka M, Davendra D (2015) Chaos driven discrete artificial bee algorithm for location and assignment optimisation problems. Swarm Evol Comput 25:15–28
21. Karaboga D, Akay B (2009) A comparative study of Artificial Bee Colony algorithm. Appl Math Comput 214(1):108–132
22. Karaboga D, Basturk B (2008) On the performance of artificial bee colony (ABC) algorithm. Appl Soft Comput 8(1):687–697
23. Yang X-S, Deb S (2010) Engineering optimisation by cuckoo search. Int J Math Model Numerical Optimiz 1(4):330–343
24. Civicioglu P, Besdok E (2013) A conceptual comparison of the Cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms. Artif Intell Rev 39(4):315–346
25. Zeng LZ, Benatallah B, Ngu AHH, Dumas M, Kalagnanam J, Chang H (2004) Qos-aware middleware for Web services composition. IEEE Trans Softw Eng 30(5):311–327
26. Alrifai M, Risse T, Nejdl W (2012) A Hybrid Approach for Efficient Web Service Composition with End-to-End QoS Constraints. ACM T Web 6(2)
27. Zhang Y, Tao F, Laili Y, Hou B, Lv L, Zhang L (2013) Green partner selection in virtual enterprise based on Pareto genetic algorithms. Int J Adv Manuf Technol 67(9-12):2109–2125
28. Xinchao Z, Boqian S, Panyu H, Zichao W, Jialei W, Yi F (2012) An improved discrete immune optimization algorithm based on PSO for QoS-driven web service composition. Appl Soft Comput 12(8):2208–2216
29. Bhandari AK, Singh VK, Kumar A, Singh GK (2014) Cuckoo search algorithm and wind driven optimization based study of satellite image segmentation for multilevel thresholding using Kapur's entropy. Expert Syst Appl 41(7):3538–3560

30. Zhang L, Guo H, Tao F, Luo YL, Si N (2010) Flexible management of resource service composition in cloud manufacturing. Paper presented at the 2010 IEEE International Conference on Industrial Engineering & Engineering Management
31. Coello CAC, Pulido GT, Lechuga MS (2004) Handling multiple objectives with particle swarm optimization. IEEE Trans Evol Comput 8(3):256–279
32. Ramacher R, Monch L (2014) Robust Multi-criteria Service Composition in Information Systems. Bus Inform Syst Eng 6(3):141–151
33. Li L, Cheng P, Ou L, Zhang Z (2010) Applying Multi-Objective Evolutionary Algorithms to QoS-Aware Web Service Composition Paper presented at the 6th International Conference on Advanced Data Mining and Applications (ADMA), Chongqing, PEOPLES R CHINA
34. Sun XY, Chen Y, Liu YP, Gong DW (2016) Indicator-based set evolution particle swarm optimization for many-objective problems. Soft Comput 20(6):2219–2232
35. Cremene M, Suciu M, Pallez D, Dumitrescu D (2016) Comparative analysis of multi-objective evolutionary algorithms for QoS-aware web service composition. Appl Soft Comput 39:124–139
36. Mirjalili S, Saremi S, Mirjalili SM, Coelho LdS (2016) Multi-objective grey wolf optimizer: a novel algorithm for multi-criterion optimization. Expert Syst Appl 47:106–119
37. Jiang QY, Wang L, Hei XH, Yu GL, Lin YY, Lu XF (2016) MOEA/D-ARA plus SBX: a new multi-objective evolutionary algorithm based on decomposition with artificial raindrop algorithm and simulated binary crossover. Knowl-Based Syst 107:197–218
38. Hemmatian H, Fereidoon A, Assareh E (2014) Optimization of hybrid laminated composites using the multi-objective gravitational search algorithm (MOGSA). Eng Optimiz 46(9):1169–1182
39. Pradhan PM, Panda G (2012) Solving multiobjective problems using cat swarm optimization. Expert Syst Appl 39(3):2956–2964
40. Patel VK, Savsani VJ (2016) A multi-objective improved teaching-learning based optimization algorithm (MO-ITLBO). Inf Sci 357:182–200
41. Akay B (2013) Synchronous and asynchronous Pareto-based multi-objective Artificial Bee Colony algorithms. J Glob Optim 57(2):415–445
42. Maximiano MD, Vega-Rodriguez MA, Gomez-Pulido JA, Sanchez-Perez JM (2013) A new Multiobjective Artificial Bee Colony algorithm to solve a real-world frequency assignment problem. Neural Comput Appl 22(7-8):1447–1459
43. Zhou J, Yao X (2016) A hybrid artificial bee colony algorithm for optimal selection of QoS-based cloud manufacturing service composition. Int J Adv Manuf Technol. doi:10.1007/s00170-016-9034-1
44. Li C, Wang S, Kang L, Guo L, Cao Y (2014) Trust evaluation model of cloud manufacturing service platform. Int J Adv Manuf Technol 75(1-4):489–501
45. Zhou J, Yao X (2016) DE-caABC: differential evolution enhanced context-aware artificial bee colony algorithm for service composition and optimal selection in cloud manufacturing. Int J Adv Manuf Technol. doi:10.1007/s00170-016-9455-x
46. Xiang F, Hu YF, Yu YR, Wu HC (2014) Qos and energy consumption aware service composition and optimal-selection based on Pareto group leader algorithm in cloud manufacturing system. Central Eur J Oper Res 22(4):663–685
47. Beloglazov A, Abawajy J, Buyya R (2012) Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing. Future Gener Comp Syst 28(5):755–768
48. Yang XS, Deb S (2014) Cuckoo search: recent advances and applications. Neural Comput Appl 24(1):169–174
49. Mantegna RN (1994) Fast, accurate algorithm for numerical simulation of Levy stable stochastic processes. Phys Rev E 49(5):4677–4683
50. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2):182–197
51. Wang YN, Wu LH, Yuan XF (2010) Multi-objective self-adaptive differential evolution with elitist archive and crowding entropy-based diversity measure. Soft Comput 14(3):193–209
52. Reynolds AM (2006) Cooperative random Levy flight searches and the flight patterns of honeybees. Phys Lett A 354(5-6):384–388
53. Zitzler E, Deb K, Thiele L (2000) Comparison of multiobjective evolutionary algorithms: Empirical results. Evol Comput 8(2):173–195
54. Deb K, Thiele L, Laumanns M (2002) Zitzler E Scalable multi-objective optimization test problems. In: Proceedings of the 2002 congress on evolutionary computation, CEC, 2002, Honolulu, HI, United states, pp 825–830
55. Zhang Q, Zhou A, Zhao S, Suganthan PN, Liu W, Tiwari S (2008) Multiobjective optimization test instances for the CEC 2009 special session and competition. University of Essex, Colchester, UK technical report
56. Nebro AJ, Luna F, Alba E, Dorronsoro B, Durillo JJ, Beham A (2008) AbYSS: Adapting scatter search to multiobjective optimization. IEEE Trans Evol Comput 12(4):439–457
57. Huang VL, Zhao SZ, Mallipeddi R (2009) Suganthan PN Multi-objective optimization using self-adaptive differential evolution algorithm, vol 2009. Trondheim, Norway
58. Nebro AJ, Durillo JJ, Nieto G, Coello CAC, Luna F, Alba E (2009) SMPSO: A new pso-based metaheuristic for multi-objective optimization. In: 2009 IEEE Symposium on computational intelligence in multi-criteria decision-making, MCDM 2009, Nashville, TN, United states, pp 66–73
59. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82
60. Beasley TM, Zumbo BD (2003) Comparison of aligned Friedman rank and parametric methods for testing interactions in split-plot designs. Comput Stat Data Anal 42(4):569–593

Jiajun Zhou was born in 1989. He received the Bachelor's degree in mechanical enginering in 2012 and is a Ph.D. candidate in the School of Mechanical and Automotive Engineering, South China University of Technology. His main research area includes cloud manufacturing, smart manufacturing, nature-inspired computing methods with applications in production and manufacturing optimization.

Xifan Yao was born in 1964. He received the Ph.D. degrees from South China University of Technology, Guangzhou, China in 1999. Now, he is a full professor at the School of Mechanical and Automotive Engineering, South China University of Technology. He is the author of 3 books and published over 200 research papers at national and international journals, as well as conference proceedings. His teaching and research interests are in the areas of production and manufacturing systems, integrated manufacturing systems, smart manufacturing, proactive manufacturing, intelligent control and scheduling in complex manufacturing systems.