Peer-to-Peer Netw. Appl. (2018) 11:1115–1128
DOI 10.1007/s12083-017-0588-y

Self-adaptive bat algorithm for large scale cloud manufacturing service composition

Bin Xu[1] · **Jin Qi**[1] · **Xiaoxuan Hu**[2] · **Kwong-Sak Leung**[3] · **Yanfei Sun**[2] · **Yu Xue**[4]

Received: 26 April 2017 / Accepted: 7 July 2017 / Published online: 14 August 2017

Abstract In order to cope with the current economic situation and the trend of global manufacturing, Cloud Manufacturing Mode (CMM) is proposed as a new manufacturing model recently. Massive manufacturing capabilities and resources are provided as manufacturing services in CMM. How to select the appropriate services optimally to complete the manufacturing task is the Manufacturing Service Composition (MSC) problem, which is a key factor in the CMM. Since MSC problem is NP hard, solving large scale MSC problems using traditional methods may be highly unsatisfactory. To overcome this shortcoming, this paper investigates the MSC problem firstly. Then, a Self-Adaptive Bat Algorithm (SABA) is proposed to tackle the MSC problem. In SABA, three different behaviors based on a self-adaptive learning framework, two novel resetting mechanisms including Local and Global resetting are designed respectively to improve the exploration and exploitation abilities of the algorithm for various MSC problems. Finally, the performance of the different flying behaviors and resetting mechanisms of SABA are investigated. The statistical analyses of the experimental results show that the proposed algorithm significantly outperforms PSO, DE and GL25.

Keywords Manufacturing service composition · Self-adaptive learning · Bat algorithm · Dual resetting

1 Introduction

IN response to the deteriorating economic situation, the fast changing market demands and the rapid rise of human costs, more and more small and medium manufacturing enterprises in China are becoming united to take risks and responsibilities together to overcome current challenges and achieve their common interests and goals. This unification means that the production resources and manufacturing capabilities of the enterprises are shared together. With the rapid development

This article is part of the Topical Collection: *Special Issue on Big Data Networking*
Guest Editors: Xiaofei Liao, Song Guo, Deze Zeng, and Kun Wang

✉ Bin Xu
xubin.njupt@foxmail.com

Jin Qi
qijin@njupt.edu.cn

Xiaoxuan Hu
comichu88@163.com

Kwong-Sak Leung
ksleung@cse.cuhk.edu.hk

Yanfei Sun
sunyanfei@njupt.edu.cn

Yu Xue
xueyu_123@nuaa.edu.cn

[1] School of Internet of Things, Nanjing University of Posts and Telecommunications, Nanjing, China

[2] School of Automation, Nanjing University of Posts and Telecommunications, Nanjing, China

[3] Chinese University of Hong Kong, Shatin, NT, Hong Kong

[4] School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing, China

of Internet of Things (IoT), cloud computing, mobile computing, big data analytics [1–9, 10] and industrial techniques [11, 12], cloud manufacturing mode (CMM) [13–18], as a new manufacturing model making the full use of this sharing, has taken up an increasing important stage. In CMM, three types of roles, including service providers, service platforms and service users, are generally included. Service providers and service users publicize and use services through the platform respectively, which generally includes three key steps [19]: (1) service providers supply the manufacturing resources and capabilities to the platform as encapsulated manufacturing services, such as material resources, equipment resources, human resources, analysis capabilities, production capabilities, and so on; (2) service users release the production demands to the platform, which in general include functional and nonfunctional requirements, the Quality of Service (QoS) indices are always used as nonfunctional indicators; and (3) the platform then analyzes the supplies and demands and makes reasonable divisions of them into small ones. Next, the platform chooses the optimal set of manufacturing services to match the different small demands and to meet the user's original demand finally. Obviously, to select the optimal manufacturing service composition becomes a very difficult problem when the homogeneous manufacturing services with very different QoS indices increase dramatically. The problem is known as a manufacturing service composition (MSC) problem [20].

Since MSC problem is NP hard [21], solving large scale MSC problems using traditional methods may be highly unsatisfactory. Since heuristic algorithms and intelligent algorithms are effective and efficient approaches for solving NP hard problems [22–26], lots of these kind of algorithms are introduced to deal with MSC problem, such as Genetic Algorithms (GAs) [27–29], Differential Evolution (DE) [30, 31], Particle Swarm Optimization (PSO) [32, 33] and so on. In these studies, MSC in different kinds of service-oriented enterprise systems is discussed. However, the scale of the MSC problem, such as the number of sub-tasks and service providers, considered is still small, which shows the limitation of these studies. In the age of the Internet, the demand for intelligent manufacturing is increasingly diversified and personalized, and the manufacturing tasks are divided into a large number of sub-tasks. At the same time, the globalization of manufacturing makes the number of service providers increasing dramatically. Therefore, it is necessary to investigate the MSC problems with large scale, which is one of the main purposes of this work.

Bat Algorithm (BA) proposed by Yang et al. in 2010 [34–36] is an optimization algorithm based on the simplified behavior and acoustics of echolocation of bats. The algorithm attracts the attention of many scholars, because of its fast convergence, less control parameters, and easier implementation. They appear to have high performance in numeric optimization. However, there are still some defects in the solution search equation of the BA, which may lead to inferior exploration and premature convergence in the exploitation process.

To improve the search abilities of BA and investigate its performance on MSC problem, a new algorithm called Self-Adaptive based Bat Algorithm (SABA) is proposed by introducing the self-adaptive learning framework into BA with multi-behaviors and dual resetting mechanisms. We perform a comprehensive comparative study of the proposed algorithm with different parameter and strategy settings. The statistical analyses of the experimental results show that the proposed algorithm achieves a better performance compared with DE, PSO and Global and local real-coded genetic algorithms based on parent-centric crossover operators (GL25) [37].

The remainder of the paper is organized as follows. Section 2 discusses the related works. Section 3 presents the MSC problem and generalizes its corresponding optimization model. Section 4 presents the details of the proposed algorithm. Section 5 presents and analyzes the experimental results to investigate the effect of different strategies and to validate our proposed approach. Finally, Section 6 is dedicated to the conclusion.

2 Related works

2.1 Manufacturing service composition problem

MSC problem is the core issue of CMM. In essence, MSC problem is another kind of service composition problem, which is similar as the web service composition problem [20]. The important difference between the web service composition problem and the MSC problem is in the participating roles. The service users and service providers are mainly the enterprises and manufactures, including not only online users, but also the combination of online and offline users in most cases in CMM, which makes the evaluation factor and evaluation mode of the MSC problem different from the web service composition problem. MSC problem has become a hot topic recently. The related research is mainly focused on formulating the model of MSC problem and solving it with a designed algorithm. In [21], considering both software and hardware cloud services in manufacturing systems, the MSC problem is modeled as a multi-objective optimization problem for the first time, which is NP hard. Then, a novel parallel intelligent algorithm is proposed to tackle the problem. Full connection topology based on coarse-grained parallelization and message passing Interface (MPI) collective communication was also adopted in the algorithm for improving the searching efficiency further. In [38], Xiang et al. established MSC model based on QoS and energy-consumption. They use group leader algorithm (GLA) to solve this problem with introducing variation strategies and a one-way and single-point crossover operator. Since the transportation from one

resource to another will inevitably increase the cost and processing time when taking into account the resources of the service chain, Lartigau et al. [39] set up MSC model by adding a geo-perspective consideration when evaluating composition. Consequently, they proposed an adapted artificial bee Colony (ABC) algorithm based on enhanced initialization to deal with this MSC problem. Weining Liu et al. [40] extended the MSC model from single-manufacturing task to multi-manufacturing tasks. That is, the platform should choose optimal cloud services composition for many users' QoS requirements on multi-functionality manufacturing tasks simultaneously. They implemented a hybrid-operator based matrix coded genetic algorithm (HO-MCGA) to tackle this problem and obtain a well performance.

The above studies addressed the different models of MSC problems. It should be noted that the scale of MSC problem in these studies is still relatively small. As a result the approaches cannot meet the demand of the current intelligent manufacturing with increasingly diversity and personality. Therefore it is necessary to investigate MSC problems with large-scale in which the number of service providers are very large and the task is divided into a large amount of sub-tasks at the same time.

2.2 Bat algorithm

BA is a new paradigm proposed by Yang et al. in 2010 [34], which imitates the echolocation behavior of bats during predation. The concept of bat algorithm is simple and easy to implement, it has gained a greater attention in numerical optimization. Pei et al. [41] integrated the BA with Differential Evolution (DE) into a new hybrid algorithm called BA-DE for solving the constrained optimization problems. Comparisons show that BA-DE outperforms most advanced methods in terms of the solution quality. Khooban et al. [42] proposed an intelligent strategy based on a combination of the bat algorithmand the Fuzzy Logic (FL) which is used to optimally tune parameters of Proportional-Integral (PI) controllers in the Automatic Generation Control (AGC). Crossover and mutation operators are introduced to the algorithm to increase the bat population diversity and an adaptive parameter updating strategy is used. Yang [43] extended the basic BA to solve multi-objective optimization problems and it has achieved efficient results in multi-objective optimization. The above works show that BA is a simple but efficient optimization algorithm compared to many other search heuristics. BA and its variants appear to have high performance in numeric optimization. However, there are still some defects in the solution search equation of the BA, which may lead to inferior exploration and premature convergence in the exploitation process.

In this paper, we are proposing three different novel behaviors in our Self-Adaptive based Bat Algorithm to improve the exploration and exploitation abilities of the algorithm for various MSC problems. Although the statistical based self-adaptive learning framework can greatly improve the performance of the intelligent algorithm, using the framework alone is still unable to avoid the premature convergence due to the loss of the population diversity during iterations of the algorithm. Therefore, we are also proposing the dual resetting mechanisms to avoid premature convergence, which is demonstrated to be very effective.

3 MSC PROBLEM FORMULATION

Let $T = \{T_1, T_2, \cdots T_i, \cdots, T_D\}$ be a complex manufacturing task where D represents the number of sub-tasks of a task. Each sub-task T_i consists of C_i candidate services S_i^j, where i represents the number of the corresponding sub-tasks, j represents the number of candidate services and $j = 1, 2, \cdots, C_i$. Each candidate service is evaluated using a QoS indicator set. By comprehensively considering the existing research results [21, 39, 40, 44, 45], the main QoS indices used in MSC problem are as follows:

1) Time t: it refers to the time consumption during the manufacturing. t includes the execution time of the service itself, the transmission time required for the remote service and the time consumed in the process of system interaction.
2) Cost c: c refers to the cost that a service user has to pay for using the capability or resource of the service. To achieve the optimal solution, c should be minimum and is calculated from the maximum price that the service demander is willing to invest independently of the currency.
3) Energy e: according to product overall lifecycle, energy consumption of a single service includes the service preparation stage, the service composition stage, the logistic transportation stage and service configuration, and the service waste disposal stage, the service using stage and the recycled energy after the cancellation of the service [44].
4) Reliability r: it assesses the ability of the service to run normally and continuously. Some of the conventional production index can be used to describe the reliability, such as the average no-failure rate and the average failure rate under a given time and condition. Obviously, service users prefer to choose the service with high reliability.
5) Maintainability m: it refers to the ability that the resource service can cope with unexpected accidents. The maintainability is the proportion of the number of the successfully processed unexpected accidents to the total number of the unexpected accidents.
6) Trust-QoS u:it refers to hold trusting entity ability in a given situation at a time slot that gives it confidence to carry a transaction out with the trusted entity. Trust-QoS is achieved by trust-QoS relationship [45].
7) Reputation v: it refers to the service credibility, which used to measure the ability of the service to achieve its

functions in the specified conditions. The evaluation of this attribute is derived from the feedback of the subjective evaluation of the users. Because this value is changed dynamically with the actual situation of the service, it is often evaluated based on the dynamic calculation method to quantify the real time reputation.

Fig. 1 depicts the main flow of the MSC carried out on the platform. It can be seen from Fig. 1 that the goal of the composition of cloud manufacturing services is to select a best path that satisfies the task demands of the service users. Theoretically, there are $\prod_{i=1}^{D} C_i$ service composition paths totally, where D and C_i are the number of sub-tasks and the number of candidate services respectively. When D and C_i increase, the scale the problem will rapidly increase and thus the problem is NP hard.

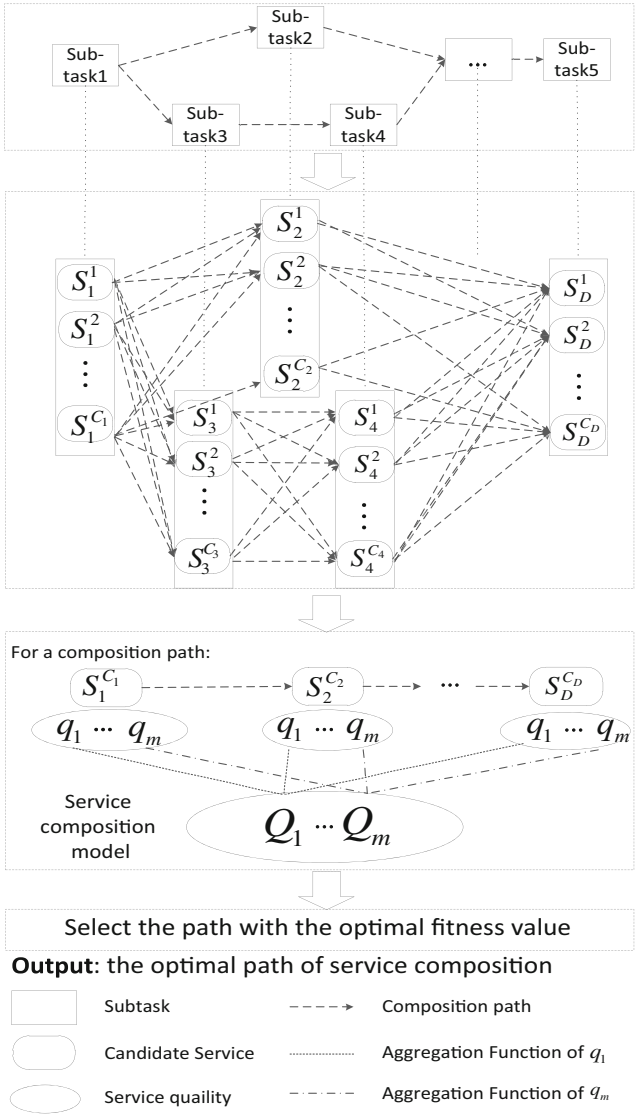

Fig. 1 The main flow of MSC

A service composition path can be expressed as $x = \{x_1, x_2, \cdots, x_i, \cdots, x_D\}$, where, x_i represents the selected candidate service to execute the ith sub-task, and $1 \le x_i \le C_i$. The cloud manufacturing service composition path in the actual scene is a directed graph showing the hybridized path structure which contains a variety of sub-structures, such as sequence, parallel, selective and circular sub-structures. Generally, the parallel, selective and circular structure can be converted into sequence structure when evaluate the path with QoS indices according to several reduction principles [33]. Thus, the MSC problem in this paper is modeled based on the sequence structure. In this structure, the QoS parameter values of each service S_i is $q(S_i^j) = \{q_1, q_2, \cdots, q_m\}$, $m = 7$, namely $q(S_i^j) = \{t, c, e, r, m, u, v\}$.

The QoS indices of the whole service composition path can be expressed as $Q = \{Q_1, \cdots, Q_m\} = \{T, C, E, R, M, U, V\}$, in which each index is evaluated by the specific rules according to the characteristics of the QoS index of the manufacturing services itself. The calculation methods of T, C, E, R, M, U, V are shown in Table 1 with the computing rules including SUM, PRODUCT and AVERAGE.

The objective of finding the optimal path is to minimize T, C, E while maximizing R, M, U, V, which is consistent with the above description of each QoS index. It is a constrained multi-objective optimization problem. In this paper, the problem is transformed into a constrained single objective optimization problem with the commonly used weighted sum method in order to facilitate the calculation. Thus, the objective function of MSC problem is as follows, which is also the fitness function used in the proposed algorithm:

$$f(x) = w_1 T(x) + w_2 C(x) + w_3 E(x) - w_4 R(x) - w_5 M(x) - w_6 U(x) - w_7 V(x) \tag{1}$$

$$\min f(x) \tag{2}$$

$$s.t. \begin{cases} q(S_i^j) \text{ satisfies each QoS index demands in MSC} \\ 1 \le j \le C_i \\ 1 \le i \le D \\ \sum_{i=1}^{D} w_i = 1 \end{cases} \tag{3}$$

Table 1 QoS calculation method in MSC

QoS index	Computing rule	Function
T, C, E	*SUM*	$\sum_{S_i^j \in x} q(S_i^j)$
R, M	*PRODUCT*	$\prod_{S_i^j \in x} q(S_i^j)$
U, V	*AVERAGE*	$\frac{1}{N} \sum_{S_i^j \in x} q(S_i^j)$ where N is the number of S_i^j in x

where $\{w_1, w_2, \ldots, w_7\}$ are the corresponding weights for each QoS index and each of them is set to the 1/7 in this paper.

4 The proposed algorithm

The proposed algorithm is detailed in subsections 4.2–4.7 as follows:

4.1 The original bat algorithm

The bats use ultrasound to detect space and foods. In the process of predation, bats can find food by adjusting flight route based on the current environmental information. In BA, the bats in the feasible region update flight speeds and adjust positions according to their own status. Each bat uses the change of the acoustic pulse frequency f_i to detect the gap between its own position and the position of the food, and then uses it to adjust the speed and direction. The new position of ith bat x_i^t and velocity v_i^t at time step t are updated by

$$f_i = f_{min} + (f_{max} - f_{min})\beta \tag{4}$$

$$v_i^t = v_i^{t-1} + \left(x_i^t - x_*\right) f_i \tag{5}$$

$$x_i^t = x_i^{t-1} + v_i^t \tag{6}$$

where $\beta \in [0, 1]$ is a random vector from a uniform distribution, and x_* is the current global best location (solution) in the population. The product $\left(x_i^t - x_*\right) f_i$ is the velocity increment used to adjust the velocity change to make an exploration over the searching space. Initially, each bat is randomly assigned a frequency which is drawn uniformly from $[f_{min}, f_{max}]$. The values of f_{min} and f_{max} are predefined depending on the domain size of the problem of interest.

The steps of the basic algorithm are as follows:

Step 1: Initialize the bat position $x_i (i = 1, 2, \cdots, N)$ and velocity v_i, define pulse frequency f_i at x_i, and initialize pulse rates r_i and the loudness A_i.
Step 2: Select individual i, adjust the frequency and update velocity to generate new solutions;
Step 3: If rand $\geq r_i$, select a solution among the best solutions, and generate a local solution around the selected best solution.
Step 4: Generate a new solution by flying randomly.
Step 5: Evaluate the solutions.
Step 6: If (rand $< A_i$ & $f(x_i) < f(x_*)$), accept the new solutions, and increase r_i and reduce A_i.
Step 7: Rank the bats and find the current best x_*.
Step 8: If the algorithm satisfies the stopping criteria, output the best individual, otherwise go to Step 2.

4.2 Self-adaptive bat algorithm

Based on the basic BA, a Self-Adaptive Bat Algorithm (SABA) is proposed to solve the MSC problem. Three novel flight behaviors are introduced below to update the position of the bat, a Local and a Global resetting mechanisms are adopted to prevent the premature convergence of the algorithm, which greatly improves the exploration and exploitation ability of the algorithm on MSC problem. The mapping between the model of BA and the MSC problem is shown in the Table 2.

Therefore, searching for an optimal service composition path to solve the MSC problem is equivalent to the process of searching for the best food source in the feasible search space in BA.

4.3 Overall algorithm procedure

The main steps of SABA are described as follows:

Step 1: Initialization: Initialize the population and the parameters in the algorithm and evaluate the fitness value of each bat in the population.
Step 2: Behavior selecting: For each bat i, generate a new position new_x_i from x_i according to the behavior selected by roulette wheel selection.
Step 3: Adaptive learning: If the new position new_x_i is better than the original x_i, then replace x_i with new_x_i, update the behavior selected probability by equation

Table 2 The mapping between the model of BA and the MSC problem

BA	MSC problem
Food source / bat location	The solution of MSC problem, that is, the service composition path. A manufacturing task can be decomposed into several sub-tasks according to the production process and requirements, each sub-task can be executed by the corresponding service selected from the candidate services set.
Fitness value of the location/ the foods source quality	The QoS quality of the service composition path
The best food source/bat location	The optimal solution
Dimensionality of the location	The number of sub-tasks

(13) and (14), then set NET_i to 0, otherwise set NET_i to $NET_i + 1$.

Step 4: Local resetting: If the NET_i reaches the predefined threshold *LIMIT*, reset all the behaviors selected probabilities of bat i to $1/M$.

Step 5: Gbest bat updating: If the generated new bat position new_x_i is better than that of Gbest bat, then replace the Gbest bat with the new one and set the *NET* value NET_{gbest} of Gbest bat to 0; otherwise, set NET_{gbest} to $NET_{gbest} + 1$.

Step 6:Global resetting: If the NET_{gbest} reaches the *LIMIT_GBEST*, the whole algorithm is reset with Gbest bat and *FEs* preserved.

Step 7:Termination: If the termination condition is satisfied, then output the current optimal solution, otherwise go to step 2.

Fig. 2 gives the framework of SABA and the details of the steps are given in the following section.

4.4 Initialization

Given that the population size is N, the foraging space of bats is D dimensional which is the same as the problem dimension. The initial population $\{x_1, x_2, \ldots, x_N\}$ is generated randomly, where x_i are the positions of the bats. The dth dimension of position x_i is generated by

$$x_i^d = \left[lb^d + \left(ub^d - lb^d\right)^* rand(0,1)\right] \tag{7}$$

where lb^d and ub^d are the lower and upper bounds of the value of dth dimension respectively. rand(0, 1) is a function which generates a random value between 0 and 1. [] represents a rounding operation. It can be seen that x_i^d is a uniformly distributed random value in the range of $[lb^d, ub^d]$.

The fitness values $f(x_i)$ are calculated by (1). The current best fitness value and the corresponding position are marked as f_{best} and x_{best} respectively, and the related bat is called Gbest bat in this paper.

4.5 Behavior selection

BA is easy to suffer from premature convergence since all the flying directions of bats are concentrated to the current best location x_{best}. To alleviate the premature convergence and enhance the exploration capability of the algorithm, multi-behaviors are designed for each bat in SABA.

The new position new_x_i of bat i is generated based on the different flight behavior $B_i (i = 1, 2, 3)$, which is chosen by the roulette wheel selection strategy according to the behavior selected probabilities of each bat. The behaviors are detailed as follows.

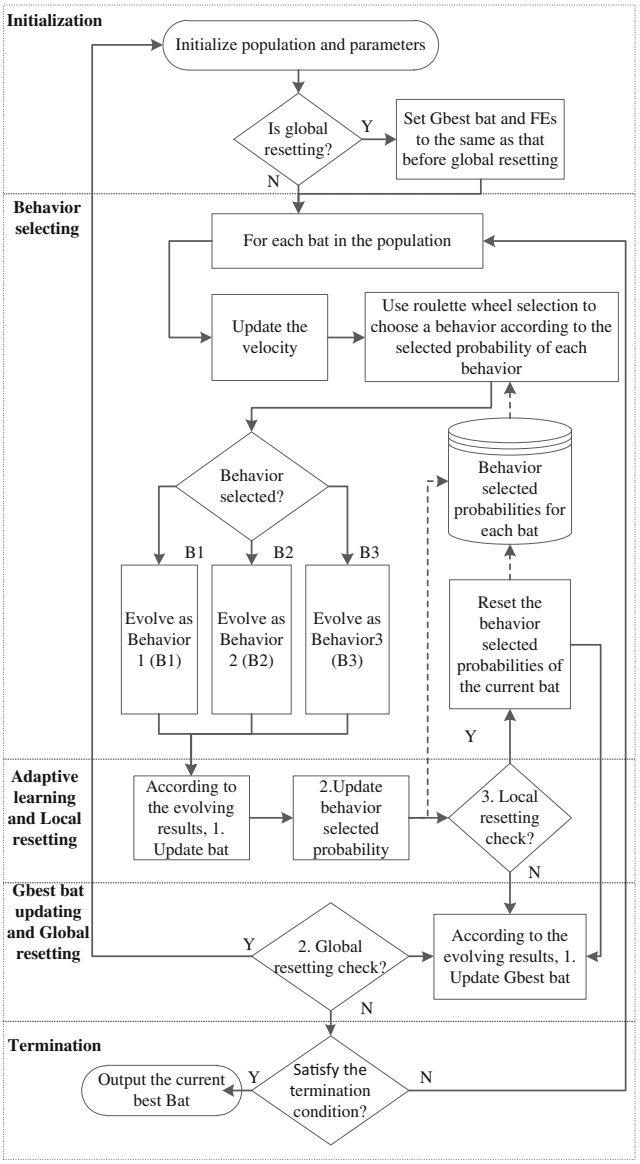

Fig. 2 The framework of SABA with Local and Global resetting

In Behavior 1 (B1), the new_x_i is generated by

$$new\_x_i = x_i + \left[F_1(x_{r1} - x_i)\right] + \left[F_2\left(x_{gbest} - x_i\right)\right] \tag{8}$$

where x_{r1} is a position randomly sampled from current bat population, $x_{r1} - x_i$ is the difference vector between x_{r1} and the current bat x_i, which makes a wider range of exploration; $x_{gbest} - x_i$ refers to another difference vector between the optimal bat location in the population and the current bat x_i, which is used to maintain the specific search direction towards optimal region. $F_1 \in [-1, 1]$ and $F_2 \in [0, 2]$ are random values drawn from a uniform distribution. [] is rounding symbol. It can be observed that two difference vectors are added to the base vector to perform perturbation so as to ensure the diversity of the population. The sum of two vectors replaces the speed vector of the original BA and can be adjusted in a

certain range by two mutation factors F_1 and F_2 to ensure the speed of convergence and avoid falling into local optima.

In Behavior 2 (B2), the dth dimension of new_x_i is generated by

$$new\_x_i^d = \left[x_{best}^d + F \times N\left(0, \left(ub^d - lb^d\right) \times rand(0,1)\right)\right] \quad (9)$$

where F is the random value of [−1, 1] satisfied the uniform distribution. [] is rounding operator. $N(0, (ub^d - lb^d)^*$ rand(0, 1)) is a value drawn from the normal distribution that mean is 0 and deviation is $(ub^d - lb^d)^*$ rand(0, 1). x_{best}^d is the dth dimension of the optimal position in the current population. It is implied from (9) that new solutions can keep certain changes in the neighborhood of optimal solution. It can not only ensure optimal direction of rapid convergence, but also avoid premature convergence.

In Behavior 3 (B3), the new_x_i is generated by

$$new\_x_i^d = \left[x_i^d + v_i^d\right] \text{if } r_i^d < A_i^d \quad (10)$$

where v_i^d is the dth dimension of the speed of bat i in the current generation. A_i^d is the loudness varies from a large value to a minimum constant value along with the iterations, $A = A_{min} + \left(1 - \frac{Current\ Iteration}{Max\ Iteration}\right) \times (A_{max} - A_{min})$, which is used to control the degree of flight direction and position. r_i^d is a random value between 0 and 1. If $r_i^d < A_i^d$, then the dth dimension should carry on the change as shown in (10), otherwise it will keep the original value. The B3 is a modified version of the flying behavior of the original BA.

For example, consider a task which has 3 sub-tasks ($D = 3$) and each sub-task has 10 candidate services, assume that $N = 5$, then the randomly generated positions and there corresponding fitness values are shown in Table 3.

In the initialization, each bat position represents a path of service composition. For example, the position of 1st bat [1, 1, 2] represents a service composition path: $S_1^1 \rightarrow S_2^1 \rightarrow S_3^2$ and the 4th bat [2, 5, 7] represents a service composition path: $S_1^2 \rightarrow S_2^5 \rightarrow S_3^7$. The velocity of all the bats is [0,0,0] initially. Obviously, the 4th bat has the optimal fitness value, so f_{best} and x_{best} are memorized to 0.2117 and [3, 3, 5] respectively.

Table 3 Example of initialization

No.	Position x	Velocity v	Fitness
1	[1, 1, 2]	[0,0,0]	0.8174
2	[2, 5, 7]	[0,0,0]	0.4229
3	[1, 4, 4]	[0,0,0]	0.6001
4	[3, 3, 5]	[0,0,0]	0.2117
5	[5, 1, 2]	[0,0,0]	0.4522
x_{best}	[3, 3, 5]	f_{best}	0.2117

4.6 Adaptive learning

The boundary control operation shown below should be performed on the new position generated by the above behaviors before evaluating.

$$\begin{cases} x_i^d = Lb \text{ if } x_i^d < Lb \\ x_i^d = Ub \text{ if } x_i^d > Ub \end{cases} \quad (11)$$

Then, the fitness value of the new position of each bat $f(new\_x_i)$ is calculated according to (1). For each bat, including Gbest bat, if the new fitness value $f(new\_x_i)$ is not better than the original fitness value $f(x_i)$, then the Not Evolve Times (NET) value NET_i of bat i is added 1, $NET_i \leftarrow NET_i + 1$. On the contrary, the following update operations are performed.

1. Update the bat. The bat i will updated to the new position new_x_i with the corresponding fitness value at same time.

$$\begin{cases} x_i \leftarrow new\_x_i \\ f(x_i) \leftarrow f(new\_x_i) \end{cases} \quad (12)$$

2. Update Behavior Selected Probability (BSP) for bat i as follows. Note that this operation is not carried out for Gbest bat, because Gbest bat is memorized to guide the evolution only.

$$BSP_i^m \leftarrow BSP_i^m + 1/M \quad (13)$$

where $m \in \{1, 2, \ldots, M\}$ is the number of selected behaviors and M is 3 in this paper. Then normalization should be carried out to ensure the selected probability $BSP_i^m \in [0, 1]$ and the sum of probabilities $\sum_{m=1}^{M} BSP_i^m = 1$.

$$BSP_i^m \leftarrow \frac{BSP_i^m}{\sum_{m=1}^{M} BSP_i^m} \quad (14)$$

3. The NET value of ith bat is set to 0, $NET(x_i) = 0$.

The behavior selected probabilities vector of each bat can be calculated and updated by the above operation. Each bat will choose the appropriate behavior by roulette wheel selection based on the behavior selected probabilities. As the behavior selected probability will be updated based on the development of the bat in the iteration, the behavior beneficial to the fitness value of the current bat is able to get more opportunities to be chosen. It is noted that, with the continuous adaptive learning, the selected probability of certain behavior may be far greater than the other behaviors, which leads to the behavior be the main behavior of the bat while others be ignored in the next iterations. As a result, the algorithm tends to fall into local optima or stop convergence when the fixed behavior is selected in successive iterations. In order to avoid

Table 4 The parameter settings of SABA, DE, PSO and GL25

Algorithm	Parameter settings
SABA	N is 50, $MaxFEs$ is 120,000, A_{min} and A_{max} are 0.1 and 0.5 respectively. $LIMIT\_GBEST$ and $LIMIT$ are 70 and 40 respectively.
SABA\LR	N is 50, $MaxFEs$ is 120,000, A_{min} and A_{max} are 0.1 and 0.5 respectively. $LIMIT\_GBEST$ is 70.
DE	This is a version provided by Rainer Storn, and the parameters are set to the same as in the code linking to http://www1.icsi.berkeley.edu/~storn/devec3.m.
PSO	This is a standard version of PSO 2007 provided in CEC2013 by Stephen Chen [46], and the parameters are set to the same as in the code linking to https://www.researchgate.net/publication/259643342_Source_code_for_an_implementation_of_Standard_Particle_Swarm_Optimization_--_revised.
GL25	The parameters are set to the same as in the original paper [37].

this situation, the Local and the Global Resetting mechanisms, which are described in the following section, are put forward to ensure the algorithm evolving continuously.

4.7 Dual resetting mechanisms

The Local and the Global resetting mechanisms are simple but effective for avoiding the algorithm falling into local optima. As seen from the algorithm flow, the objective of Local resetting and Global resetting are different. The former is for each bat, it performs as follows: if the *NET* value of the bat exceeds the preset threshold, then reset all the behaviors selected probabilities of this bat to $1/M$. The latter is for Gbest bat only, it performs as follows: if the *NET* value of Gbest bat exceeds the preset threshold, then reset the algorithm itself with Gbest bat and the number of Fitness Evaluations (*FEs*) preserved. Resetting algorithm means that all the parameters are reinitialized and the algorithm is restarted. It is implied that, when the behavior cannot continuously optimize the fitness value for the individual bat, Local resetting will be carried out to make the subsequent behavior with randomization, so as to provide a possibility for the current individual to escape from local optimum. Global resetting is focused on the overall behavior of the bat population in the algorithm, which helps to avoid stopping convergence because of the lack of the diversity of the whole population. In general, the Local and the Global resetting mechanisms are self-recovery mechanisms when the algorithm is prematurely converged.

5 Experiments and analysis

In most research works on MSC problem [21, 33], because of the lack of public data sets, the synthetic datasets are always used for simulation. These data sets are generated with a hypothesis, namely, the number of sub-tasks in the manufacturing process is relatively small. However, with the increasing of the degree of production refinement, and the expansion of the demands for personalized customization manufacturing based on the Internet, both the number of manufacturing sub-tasks, and the number of service candidates, are growing dramatically. Therefore, it is necessary to test the performance of the algorithm on larger scale data sets. In this paper, the data set is constructed from small scale to large scale, which is used to test the algorithm in a variety of conditions. The values of QoS properties of each service are generated randomly from (0,1). The number of sub-tasks D and service candidates C varies from small to large, which consists of two cases: (1) D fixed to 100, C increased from 500 to 5000; (2) C fixed to 500, D increased from 100 to 1000.

Table 5 The mean and standard deviation of fitness values of SABA, DE, PSO, GL25, SABA\LR, SABA\LR, SABA\GR, SABA\R, SABA\SAL and RBA on MSC problem with D fixed to 100, C varying from 500 to 5000

	D	100	100	100	100	100	100	100	100	100	100
	C	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
SABA	Mean	**12.2**	**12.18**	**11.96**	**12.04**	**11.91**	**12**	11.92	**11.94**	**12.09**	**12.13**
	Std	0.31	0.28	0.28	0.29	0.27	0.32	0.32	0.25	0.29	0.23
DE	Mean	16.34	16.45	16.4	16.5	16.49	16.57	16.56	16.53	16.51	16.54
	Std	0.24	0.18	0.22	0.17	0.18	0.15	0.16	0.19	0.19	0.18
PSO	Mean	16.85	17.11	16.97	17.14	16.76	17.15	16.85	17.14	17.16	17.11
	Std	0.33	0.35	0.36	0.36	0.35	0.48	0.39	0.27	0.32	0.34
GL25	Mean	18.35	18.37	18.42	18.37	18.41	18.4	18.44	18.43	18.41	18.41
	Std	0.11	0.13	0.08	0.13	0.13	0.13	0.1	0.1	0.13	0.14
SABA\LR	Mean	12.35	12.34	12.13	12.21	12.06	12.2	**11.87**	12.24	12.25	12.2
	Std	0.29	0.27	0.3	0.31	0.27	0.24	0.23	0.26	0.29	0.27
SABA\GR	Mean	13.76	13.78	13.6	13.59	13.34	13.53	13.29	13.57	13.63	13.79
	Std	0.36	0.27	0.36	0.35	0.25	0.37	0.4	0.29	0.34	0.31
SABA\R	Mean	13.22	13.5	13.27	13.39	13.11	13.38	12.96	13.17	13.24	13.48
	Std	0.35	0.36	0.27	0.35	0.35	0.41	0.38	0.45	0.39	0.37
SABA\SAL	Mean	19.02	19.34	18.87	19.16	18.34	19.21	18.57	18.99	19.11	19.05
	Std	0.34	0.24	0.4	0.34	0.41	0.35	0.33	0.42	0.37	0.24
RBA	Mean	14.01	13.93	13.43	13.67	13.28	13.54	13.23	13.35	13.6	13.7
	Std	0.51	0.28	0.39	0.5	0.28	0.47	0.37	0.36	0.37	0.42

Table 6 The mean and standard deviation of fitness values of SABA, DE, PSO, GL25, SABA\LR, SABA\LR, SABA\GR, SABA\R, SABA\SAL and RBA on MSC problem with *C* fixed to 500, *D* varying from 100 to 1000

	D	100	200	300	400	500	600	700	800	900	1000
	C	500	500	500	500	500	500	500	500	500	500
SABA	Mean	**12.2**	**29.05**	**47.37**	**67.08**	**87.07**	**105.48**	124.8	144.7	164.78	186.59
	Std	0.31	0.46	0.56	1.03	1.33	1.73	1.61	2.45	2.71	2.95
DE	Mean	16.34	35.75	55.7	75.88	96.13	116.31	136.81	157.42	178.02	198.84
	Std	0.24	0.25	0.36	0.37	0.41	0.54	0.41	0.51	0.49	0.68
PSO	Mean	16.85	35.84	55.94	76.51	97.19	117.27	137.32	157.77	178.39	199.16
	Std	0.33	0.53	0.68	0.85	0.85	1.01	1.16	1.26	1.21	1.17
GL25	Mean	18.35	38.61	58.99	79.72	100.44	121.3	142.14	163.01	183.87	204.88
	Std	0.11	0.13	0.2	0.3	0.24	0.31	0.43	0.34	0.47	0.33
SABA\LR	Mean	12.35	29.11	47.42	67.17	87.18	106.2	**123.62**	**144.45**	**164.28**	**184.56**
	Std	0.29	0.4	0.68	1.06	1.66	2.02	1.33	1.83	2.14	3.23
SABA\GR	Mean	13.76	30.97	50.17	69.87	89.44	108.73	127.73	147.78	167.26	188.48
	Std	0.36	0.38	0.7	0.89	0.99	0.98	1.1	1.7	1.59	2.49
SABA\R	Mean	13.22	30.65	48.88	68.28	87.74	106.56	125.32	144.97	165.08	185.16
	Std	0.35	0.82	0.78	1.16	1.47	2.1	1.89	2.23	2.6	2.21
SABA\SAL	Mean	19.02	38.97	60.15	81.39	102.51	122.89	143.5	164.42	185.62	206.88
	Std	0.34	0.51	0.65	0.53	0.77	0.84	0.99	1.23	1.34	1.33
RBA	Mean	14.01	31.33	51.2	71.85	91.77	111.28	129.76	149.94	171.48	190.86
	Std	0.51	0.64	0.9	1.54	1.98	1.88	2.33	2.73	2.74	3.83

5.1 Results analysis and comparison with other similar algorithms

In order to verify the effectiveness of the proposed algorithm, SABA and SABA without Local resetting (SABA\LR) were compared with other commonly used algorithms including DE, PSO and GL25, which are available as open source. The settings of MSC problem are as follows: (1) *D* was set to 100, *C* varied from 500 to 5000 and (2) *D* increased from 100 to 1000, *C* was set to 500. For each algorithm, 30 independent runs were conducted with $MaxFEs = 120000$ as the termination criterion and the population size $N = 50$. The other parameter settings are descripted in Table 4.

The individuals in DE, PSO and GL25 were the discretization form by rounding operator because the MSC problem is a discrete optimization problem. The mean and standard deviation results of these algorithms are shown in Tables 5 and 6. Tables 7 and 8 are Wilcoxon's rank sum test at a significance level of 0.05, in which "+", "−" and "≈" denote that the performance of corresponding algorithm is better than, worse than, and similar to that of SABA, respectively. It can be seen that SABA and SABA\LR provide the best results for all the problems. Even though the stability of SABA and SABA\LR becomes gradually worse along with the increase of the problem dimension, the performance of SABA\LR in terms of solution quality is rather competitive. This may be due to the great exploration and exploitation ability of multi-behaviors based on the self-adaptive learning framework and the resetting mechanisms especially Global resetting.

Furthermore, the convergence maps of SABA, SABA\LR, DE, PSO and GL25 on MSC problem with different conditions in Fig. 3 show that the SABA and SABA\LR always converge faster than other algorithms. It can be observed that DE, PSO and GL25 suffer from local optima while SABA and SABA\LR have better global search ability. The reason for the results is that SABA and SABA\LR are able to select most appropriate behavior at different stages for different conditions by self-adaptive learning framework and keeping diversity of the population by resetting mechanism. Therefore, the proposed algorithms not only have good convergence efficiency, but also maintain well search ability when the other algorithms are converging slowly or even prematurely.

5.2 Analysis of dual resetting

The experiment was designed to investigate the effect of the resetting mechanisms on the performance of the

Table 7 Wilcoxon's rank sum test results of SABA, DE, PSO, GL25 and SABA\LR on MSC problem with D fixed to 100, C varying from 500 to 5000

	DE	PSO	GL25	SABA\LR
−	10	10	10	6
+	0	0	0	1
≈	0	0	0	3

Table 8 Wilcoxon's rank sum test results of SABA, DE, PSO, GL25 and SABA\LR on MSC problem with *C* fixed to 500, *D* varying from 100 to 1000

	DE	PSO	GL25	SABA\LR
−	10	10	10	0
+	0	0	0	2
≈	0	0	0	8

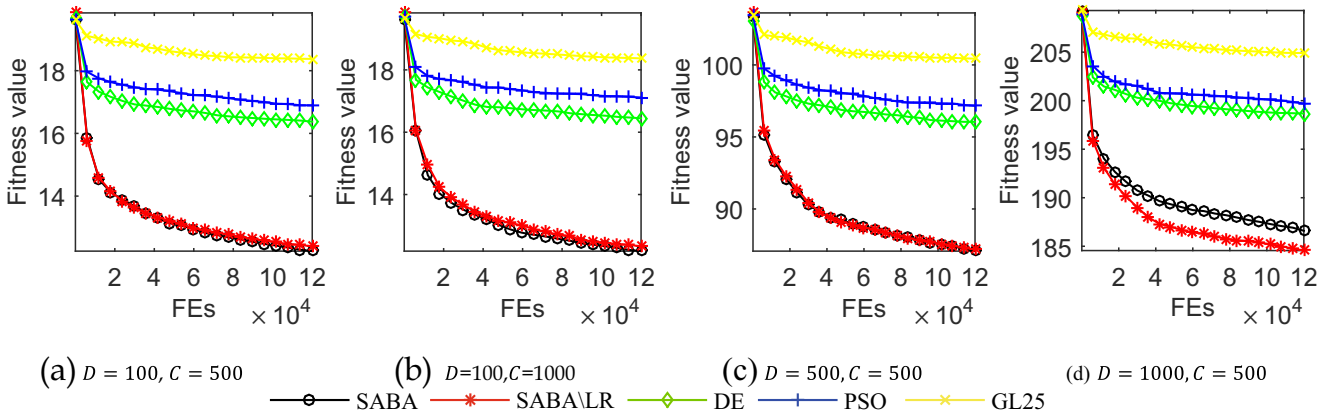

Fig. 3 The convergence characteristics of SABA, DE, PSO and GL25 on MSC problem where FEs varies from 50 to 120,000

algorithm. SABA without Local resetting (SABA\LR), SABA without Global resetting (SABA\GR), SABA without resetting including both Local and Global resetting (SABA\R) were compared with SABA on MSC problem. The parameter settings of SABA, SABA\LR, SABA\GR and SABA\R are shown in Table 9.

Table 5 reports the mean and standard deviation of fitness values by applying the four algorithms to optimize MSC problem, in which D is fixed to 100, C varies from 500 to 5000. Table 6 shows the mean and standard deviation of fitness values of SABA, SABA\LR, SABA\GR and SABA\R on MSC problem with C fixed to 500, D varying from 100 to 1000. The best results are typed in bold. The data are the statistical results obtained from 30 independent runs. As shown in Tables 5 and 6, SABA and SABA\LR outperform the SABA\GR and SABA\R in all cases. For cases, $D = 100$ and $C = 3500$, $D = 700$ and $C = 500$, $D = 800$ and $C = 500$, $D = 900$ and $C = 500$, $D = 1000$ and $C = 500$, SABA\LR has a better performance than SABA. While for other cases, the solution quality of SABA is higher than that of SABA\LR. The fact that SABA\R works better than SABA\GR demonstrates that the use of the Local resetting mechanism alone has no positive effect on the performance of the algorithm. From the above results we can observe that the resetting mechanism especially the Global resetting mechanism is really effective for the algorithm to improve the solution quality.

5.3 Analysis of self-adaptive learning framework

This experiment was designed to study the effects of self-adaptive learning framework on algorithm performance when running SABA on MSC problem. SABA without Self-Adaptive Learning (SABA\SAL), Random Bat Algorithm (RBA), are compared with SABA on MSC problem. In SABA\SAL, only the B3 was used for updating the position and velocity of the bat. In RBA, three behaviors were selected randomly instead of self-adaptive learning, that is, the probabilities of the behaviors were not considered in the process of the algorithm. The parameter settings of SABA are the same as in 5.1, and the other parameter settings of SABA\SAL and RBA are the same as in SABA.

Tables 5 and 6 presents the mean and standard deviation results of these algorithms with 30 independent runs. It can be seen that SABA performs best out of the three algorithms. Comparing with SABA\SAL, RBA obtains better results on all conditions. It is because the multi-behaviors can help the algorithm having a better ability of exploration than the single behavior. From the comparison results of the three algorithms, we can investigate that SABA can not only maintain the global search ability, but also have the ability of adopting different behaviors in different stages of the algorithm based on self-adaptive learning, which makes the algorithm perform best.

Furthermore, Fig. 4 illustrates the average selected probability of each behavior of all bats in the process of

Table 9 The parameter settings of SABA, SABA\LR, SABA\GR and SABA\R

Algorithm	Parameter settings
SABA	N is 50, *MaxFEs* is 120,000, A_{min} and A_{max} are 0.1 and 0.5 respectively. *LIMIT_GBEST* and *LIMIT* are 70 and 40 respectively.
SABA\LR	N is 50, *MaxFEs* is 120,000, A_{min} and A_{max} are 0.1 and 0.5 respectively. *LIMIT_GBEST* is 70.
SABA\GR	N is 50, *MaxFEs* is 120,000, A_{min} and A_{max} are 0.1 and 0.5 respectively. *LIMIT* is 40.
SABA\R	N is 50, *MaxFEs* is 120,000, A_{min} and A_{max} are 0.1 and 0.5 respectively.

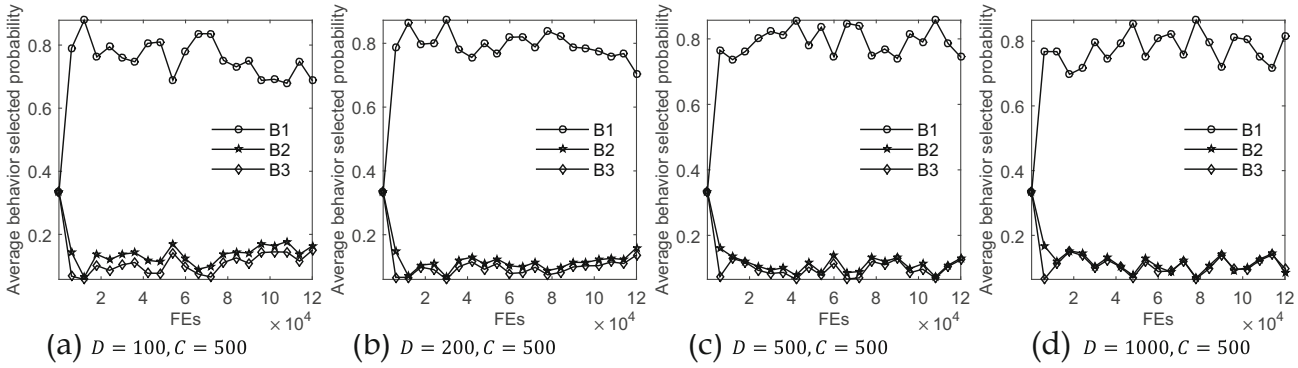

Fig. 4 The average behavior selected probabilities of all bats of SABA running 30 times independently, where FEs varies from 50 to 120,000

SABA for four MSC problem conditions. Fig. 4a represents the average selected probabilities of three behaviors in the whole population under FEs increasing from 50 to 120,000, where the number of tasks $D = 100$ and the number of candidate services $C = 500$. As can be seen from the figure, at the beginning, the selected probability of each behavior is 1/3 because of the equal selection probability of three behaviors in the algorithm initialization. Throughout the iterative process, the selected probability of B1 is significantly higher than the other two behaviors, which implies B1 plays a dominant role in the solution process of the algorithm on this problem. Although the selected probability of B1 is relatively high in overall trend, there is a certain degree of fluctuation in the whole iterative process. The result is consistent with the self-adaptive learning framework in which different behaviors in different iterative stages have different effects of the algorithm to solve the different problems. Similarly, in the case of the D=200, D=500 and D=1000, B1 shows its significant effect for solving this problem than the other two behaviors. The effect of the resetting mechanisms cannot be clearly demonstrated in the figures since the figures only show the average results.

Therefore, Fig. 5 illustrates the selected probability of each behavior of a bat of the median run of SABA for four MSC problem conditions. As can be seen from the Fig. 5a, the selected probability of each behavior is the same and equal to 1/3. Different from the previous analysis of the average probability, the three behaviors have a sharp increase or decrease since the behavior is always changing for the individual. If the B2 generates a better solution in one generation, the selected probability of B2 behavior is increased in the next generation, while the selected probabilities of the others are fall. It can be found that when the FEs equals to 40,000, the selected probability point of three kinds of behaviors will coincide again, which may be due to the resetting mechanisms of the bat (Local resetting) or the algorithm (Global resetting). Similar results can be observed in other graphs. The red dot in the figure indicates the actual choice of the behavior in the algorithm. As can be seen in Fig. 5, the behavior with the largest selected probability is always being chosen. The other behaviors with low selected probabilities also have the chance to be selected because of the randomness of roulette wheel selection. For example, B1 has a largest selected probability when FEs = 90,000 in Fig. 5a, but in fact B3 is selected. This shows that the behavior of each bat is not entirely determined

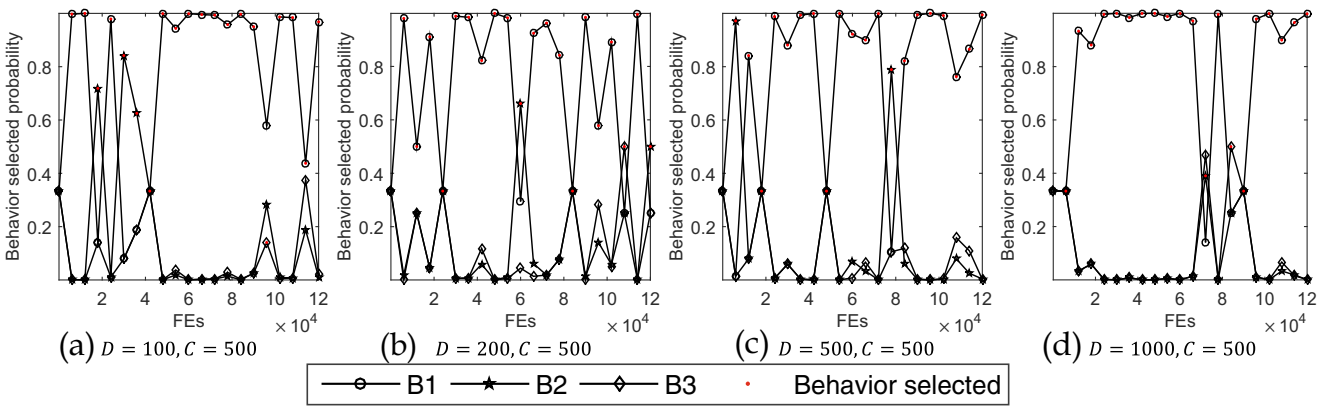

Fig. 5 The behavior selected probability and the real selected behavior of a specific bat of the median run of SABA, where FEs varies from 50 to 120,000

by the selected probability and the choice of the behavior has some randomness on the basis of probability, which brings a perturbation to some extent.

5.4 Discussion

The experiments show that SABA performs better in terms of solution quality and convergence speed than the other algorithms on MSC problems, especially when the problem scale becomes larger. The main reason is that SABA has some good characteristics compared with other algorithms. Firstly, SABA has a multi-behavior selection framework based on self-adaptive learning. In this framework, three behaviors have different properties and search abilities. For example, in B1, inspired by DE algorithm, two difference vectors, which express the promoting and stochastic direction respectively, are added to the current individual. This leads to the current individual moving to the promoting area with certain randomness. B1 is demonstrated to be helpful to provide a good balance of global and local search capability for the algorithm, as shown in 5.4. B2 is inspired from EDA [47] and ES [48] algorithms. It uses the Gauss distribution in the solution space of the problem to carry out the sampling and the updating operations, which enables good exploitation. As for B3, it is similar to that in the original bat algorithm. The main difference is that the updating dimensions in B3 are selected according to the parameter A which gradually decreases as the number of iterations increases. Self-adaptive framework helps the algorithm to select the most appropriate behavior in different stages according to different problems, thus enhances the dynamics and universality of the algorithm. Furthermore, SABA has dual resetting mechanisms, namely, Local resetting and Global resetting. Local resetting can make the individual behavior not tend to be single, while Global resetting can help escape from local optima. Both of them are able to enhance the diversity of the population. It is worth noting that Global resetting is based on the reservation of the global best solution to avoid the full blindness of the subsequent iteration process and to ensure that the promising area in the search space will not be discarded. Experiments in section 5.2 show that this mechanism can significantly improve the performance of the algorithm.

6 Conclusion

With the rapid growth of manufacturing services in CMM, it has become an important issue to obtain the optimal manufacturing service composition with a large number of service providers and subtasks. We have proposed a self-adaptive based bat algorithm (SABA) to tackle the MSC problem. In SABA, the self-adaptive learning framework with three different novel behaviors is designed to make a tradeoff between the global exploration and the local exploitation of the algorithm. Additionally, the Local and the Global resetting mechanisms are introduced into the algorithm to avoid suffering from premature convergence. The proposed algorithm has advantages in terms of the solution quality and efficiency compared with other algorithms especially for large-scale testing instances. Therefore, it is a competitive solution for MSC problem.

In the future work, we will obtain more practical data from the industrial field for experiments, and further verify the value of this method in real world. In addition, we will also perform further research on multiple behaviors, including how to design evolutionary behaviors, combine behaviors, and choose the quantity of behaviors.

Acknowledgements This paper was supported by Natural Science Foundation of China (61572262), Natural Science Foundation of Jiangsu Province of China (No. BK20160910, BK20141427), Natural science fund for colleges and universities in Jiangsu Province (No. 16KJB520034), NUPTSF (Grant Nos. NY213047, NY213050, NY214102, NY214098), A Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD), Jiangsu Collaborative Innovation Center on Atmospheric Environment and Equipment Technology (CICAEET).

References

1. Wang K, Shao Y, Shu L, Zhu C, Zhang Y (2016) Mobile big data fault-tolerant processing for ehealth networks. IEEE Netw 30(1):36–42
2. Xia Z, Wang X, Sun X, Wang Q (2016) A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data. IEEE Trans Parallel Distrib Syst 27(2):340–352
3. Wang K, Shao Y, Shu L, Han G, Zhu C (2015) LDPA: a local data processing architecture in ambient assisted living communications. IEEE Commun Mag 53(1):56–63
4. Liu Q, Cai W, Shen J, Fu Z, Liu X, Linge N (2016) A speculative approach to spatial-temporal efficiency with multi-objective optimization in a heterogeneous cloud environment. Secur Commun Netw 9(17):4002–4012
5. Wang K, Mi J, Xu C, Zhu Q, Shu L, Deng D-J (2016) Real-Time Load Reduction in Multimedia Big Data for Mobile Internet. ACM Trans Multimed Comput Commun Appl 12(5):76
6. Fu Z, Sun X, Liu Q, Zhou L, Shu J (2015) Achieving efficient cloud search services: multi-keyword ranked search over encrypted cloud data supporting parallel computing. IEICE Trans Commun E98B(1):190–200
7. Wang K, Wang Y, Sun Y, Guo S, Wu J (2016) Green industrial internet of things architecture: an energy-efficient perspective. IEEE Commun Mag 54(12):48–54
8. Fu Z, Wu X, Guan C, Sun X, Ren K (2016) Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement. IEEE Trans Inf Forensics Secur 11(12):2706–2716
9. Wang K, Qi X, Shu L, Deng DJ, Rodrigues JJPC (2016) Toward trustworthy crowdsourcing in the social internet of things. IEEE Wirel Commun 23(5):30–36

10. Wang K, Wang Y, Zeng D, Guo S (2017) An SDN-based architecture for next-generation wireless networks. IEEE Wirel Commun 24(1):25–31
11. Wang K, Zhuo L, Shao Y, Yue D, Tsang KF (2016) Toward distributed data processing on intelligent leakpoints prediction in petrochemical industries. IEEE Trans Ind Inf 12(6):2091–2102
12. Wang K, Lu H, Shu L, Rodrigues JJPC (2014) A context-aware system architecture for leak point detection in the large-scale petrochemical industry. IEEE Commun Mag 52(6):62–69
13. Li B, Zhang L, Wang S, Tao F, Cao J, Jiang X, Song X, Chai X (2010) Cloud manufacturing: a new service-oriented networked manufacturing model. Comput Integr Manuf Syst 16(1):1–7
14. Guo L (2016) A system design method for cloud manufacturing application system. Int J Adv Manuf Technol 84(1–4):275–289
15. Xu Y, Chen G, Zheng J (2016) An integrated solution-KAGFM for mass customization in customer-oriented product design under cloud manufacturing environment. Int J Adv Manuf Technol 84(1–4):85–101
16. Liu K, Zhong P, Zeng Q, Li D, Li S (2017) Application modes of cloud manufacturing and program analysis. J Mech Sci Technol 31(1):157–164
17. Qiu X, He G, Ji X (2016) Cloud manufacturing model in polymer material industry. Int J Adv Manuf Technol 84(1–4):239–248
18. 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
19. Liu I, Jiang H (2012) Research on key technologies for design services collaboration in cloud manufacturing. In proceedings of the 2012 I.E. 16th international conference on computer supported cooperative work in design (CSCWD), pp 824–829
20. Fu C, Xiao M (2014) Optimization method of cloud service composition in cloud manufacturing environment. Appl Res Comput 31(6):1744–1747
21. 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
22. Wang H et al (2017) Firefly algorithm with neighborhood attraction. Inf Sci 382:374–387
23. Wang H, Wang W, Sun H, Rahnamayan S (2016) Firefly algorithm with random attraction. Int J Bio-Inspired Comput 8(1):33–41
24. Shao Y, Wang K, Shu L, Deng S, Deng D-J (2016) Heuristic optimization for reliable data congestion analytics in crowdsourced eHealth networks. IEEE Access 4:9174–9183
25. 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
26. Xue Y, Jiang J, Zhao B et al (2017) A self-adaptive artificial bee colony algorithm based on global best for global optimization. Soft computing: 1-18
27. Feng J, Kong L (2015) A fuzzy multi-objective genetic algorithm for QoS-based cloud service composition. 2015 11th 2015 11th international conference on semantics, knowledge and grids (skg), pp 202–206
28. Li Y, Yao X, Zhou J (2016) Multi-objective optimization of cloud manufacturing service composition with cloud-entropy enhanced genetic algorithm. Stroj Vestn-J Mech E 62(10):577–590
29. Gupta IK, Kumar J, Rai P (2015) Optimization to quality-of-service-driven web service composition using modified genetic algorithm. 2015 international conference on computer, communication and control (ic4)
30. C. Liu, X. Xiang, C. Zhang, and L. Zheng, A Decision Model for Berth Allocation Under Uncertainty Considering Service Level Using an Adaptive Differential Evolution Algorithm. Asia-Pacific Journal of Operational Research, 33(6):1650049, Dec. 2016
31. Zhou Y, Zhang C, Zhang B (2015) Multi-objective service composition optimization using differential evolution. 2015 11th international conference on natural computation (icnc), pp 233–238
32. Hossain MS, Moniruzzaman M, Muhammad G, Ghoneim A, Alamri A (2016) Big data-driven service composition using parallel clustered particle swarm optimization in mobile environment. IEEE Trans Serv Comput 9(5):806–817
33. Tao F, Zhao D, Hu Y, Zhou Z (2008) Resource service composition and its optimal-selection based on particle swarm optimization in manufacturing grid system. IEEE Trans Ind Inf 4(4):315–327
34. X.-S. Yang, A New Metaheuristic Bat-Inspired Algorithm. In Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), J. R. González, D. *A. pelta*, *C. Cruz*, G. Terrazas, and N. Krasnogor, Eds. Springer, Berlin Heidelberg, pp 65–74, 2010
35. Nakamura RYM, Pereira LAM, Costa KA, Rodrigues D, Papa JP, Yang XS (2012) BBA: A Binary Bat Algorithm for Feature Selection. In 2012 25th SIBGRAPI conference on graphics, Patterns and Images, pp 291–297
36. Senthilnath J, Kulkarni S, Benediktsson JA, Yang XS (2016) A novel approach for multispectral satellite image classification based on the bat algorithm. IEEE Geosci Remote Sens Lett 13(4):599–603
37. García-Martínez C, Lozano M, Herrera F, Molina D, Sánchez AM (2008) Global and local real-coded genetic algorithms based on parent-centric crossover operators. Eur J Oper Res 185(3):1088–1113
38. Xiang F, Hu Y, Yu Y, Wu H (2013) QoS and energy consumption aware service composition and optimal-selection based on Pareto group leader algorithm in cloud manufacturing system. CEJOR 22(4):663–685
39. Lartigau J, Xu X, Nie L, Zhan D (2015) Cloud manufacturing service composition based on QoS with geo-perspective transportation using an improved artificial bee Colony optimisation algorithm. Int J Prod Res 53(14):4380–4404
40. Liu W, Liu B, Sun D, Li Y, Ma G (2013) Study on multi-task oriented services composition and optimisation with the 'multi-composition for each task' pattern in cloud manufacturing systems. Int J Comput Integr Manuf 26(8):786–805
41. Shengyu Pei, Aijia Ouyang, and Lang Tong (2015) A Hybrid Algorithm Based on Bat-Inspired Algorithm and Differential Evolution for Constrained Optimization Problems. Int J Patt Recogn Artif Intell 29: 1559007
42. Khooban MH, Niknam T (2015) A new intelligent online fuzzy tuning approach for multi-area load frequency control: self adaptive modified bat algorithm. Int J Electr Power Energy Syst 71:254–261
43. Yang X-S (2012) Bat algorithm for multi-objective optimisation. Int J Bio-Inspired Comput 3(5):267–274
44. Tao F, Hu Y, Zhao D, Zhou Z, Zhang H, Lei Z (2008) Study on manufacturing grid resource service QoS modeling and evaluation. Int J Adv Manuf Technol 41(9–10):1034–1042
45. Tao F, Hu YF, Zhou ZD (2009) Application and modeling of resource service trust-QoS evaluation in manufacturing grid system. Int J Prod Res 47(6):1521–1550
46. J. Montgomery, Stephen Chen (2014) Standard Particle Swarm Optimization on the CEC2013 Real-Parameter Optimization Benchmark Functions – revised
47. Mark H, Martin P (2011) An introduction and survey of estimation of distribution algorithms. Swarm and Evolutionary Computation 1(93):111–128
48. Hansen N, Ostermeier A (2001) Completely Derandomized self-adaptation in evolution strategies. Evol Comput 9(2):159–195

Bin Xu received the M.S. and Ph.D. degrees in Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2007 and 2011, respectively. He is currently a Lecturer in Nanjing University of Posts and Telecommunications, Nanjing, China. His research interests are mainly in evolutionary computation, service-oriented manufacturing, and Internet of Things.

Jin Qi received the B.S. degree in computer science and technology from Liaocheng University, Liaocheng, China, in 2005. He received the M.S. degrees in computer application from Jiangnan University, Wuxi, China, in 2008. He received the Ph.D. degree in information network from Nanjing University of Posts and Telecommunications, Nanjing, China, in 2012. He is currently an Associate Professor at School of Internet of Things in Nanjing University of Posts and Telecommunications. His main research interests are in Intelligent Computing, Industrial Internet, and Internet of Things.

Xiaoxuan Hu received the B.S. degree in Automation from Nanjing University of Posts and Telecommunications, Nanjing, China, in 2014. She studies in the Nanjing University of Posts and Telecommunications for the Ph.D. degree in pattern recognition and intelligent system since September 2014. Her main research interests are in intelligent computing, future network, and Internet of Things and energy Internet.

Kwong-Sak Leung (M'77-SM'89) received the BSc (Eng.) and PhD degrees from the University of London, Queen Mary College, in 1977 and 1980, respectively. He joined the Computer Science and Engineering Department at the Chinese University of Hong Kong in 1985, where he is currently a professor of computer science & engineering. His research interests are in bioinformatics and soft computing including evolutionary computation, parallel computation, probabilistic search, information fusion and data mining, fuzzy data and knowledge engineering. He is a senior member of the IEEE.

Yanfei Sun received the Ph.D. degrees in information network from Nanjing University of Posts and Telecommunications, Nanjing, China, in 2006. He is currently a Professor at School of Internet of Things in Nanjing University of Posts and Telecommunications. His main research interests are in intelligent optimization, network management, machine learning, and future network.

Yu Xue received the Ph.D. degree in Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2013. He is currently an Associate Professor in Nanjing University of Information Science and Technology, Nanjing, China. His research interests are mainly in evolutionary computation and swarm intelligence. He is a member of IEEE.