Journal of Intelligent Manufacturing
https://doi.org/10.1007/s10845-022-02032-w

Particle swarm optimization service composition algorithm based on prior knowledge

Hongbin Wang[1,2] · **Yang Ding**[1,2] · **Hanchuan Xu**[3]

Received: 11 December 2021 / Accepted: 16 September 2022

Abstract
In order to quickly find an appropriate composition of services that meet the individual user's requirements in the Internet big data, this paper proposes an improved particle swarm service composition method based on prior knowledge. This method firstly mines the service composition partial segments with certain frequencies of usage from a large number of historical service composition solutions, i.e. the service pattern. While receiving the user's service composition requirement, this method uses the service pattern matching algorithm proposed in this paper to match the corresponding service patterns as a partial solution of this composition requirement. Then the method proposes an improved particle swarm algorithm for the part that do not successfully match the corresponding service patterns. This improved particle swarm algorithm has a mechanism to escape from the local optima. Finally, the method integrates the partial solutions of the two aspects into a complete solution, i.e. a complete service composition solution. This paper compares the optimality, time complexity and convergence with other related service composition optimization algorithms through simulation experiments. According to the analysis of the experimental results, the method proposed in this paper shows good performance in three aspects: optimality, time complexity and convergence.

Keywords Service composition · Service pattern · Particle swarm algorithm · Quality of service

Introduction

With the development of cloud computing, IoT technologies (Sailer, 2014) and IoD (Abualigah, 2021), offline services are more and more employed online by virtualization technology to cooperate with online services (Mladen, 2008; Zhang, 2020). The service requirements grew with the increasing complexity, which formed the IoS (Momeni, 2021), Big service (Zhang, 2020), and cross-border services (Falch, 2020). Therefore, service composition is required in order to meet the complex needs of users. The first challenge is to quickly and efficiently construct a service composition plan via composition technology to face the complex large number of available services. Since the single web service often failed to meet customer needs properly, the service composition becomes urgent (Klai, 2016). During the service composition, each type of service has numerous candidate services with the same function but different Quality of Service (QoS). The QoS-aware service composition (QoS-SC) attracts both academic and industrial attention recently. Current Web composition methods are mostly based on QoS-aware Web service composition methods. There are various solutions to the QoS-aware Web service composition optimization problems, which can be divided into three categories (Huo et al., 2015):

Hongbin Wang and Yang Ding have contributed equally to this work.

✉ Hanchuan Xu
xhc@hit.edu.cn

Hongbin Wang
whbin2007@126.com

Yang Ding
dy441326@gmail.com

[1] Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Kunming 650500, Yunnan, China

[2] Yunnan Key Laboratory of Artificial Intelligence, Kunming University of Science and Technology, Kunming 650500, Yunnan, China

[3] Faculty of Computing, Harbin Institute of Technology, Harbin 150001, Heilongjiang, China

Published online: 01 October 2022

(1) The local search method, different QoS attribute values are mapped to a single index through an aggregation function, and specific service is selected from each candidate service set with reference to multiple indexes to form a service composition plan. This method is only effective for compositions with a small number of QoS attribute restrictions but not for those with large numbers (Cook, 2021; Maatouk et al., 2021).
(2) The global optimization method transforms the entire service composition into a hybrid linear programming problem with the solution. It has certain restrictions on the algorithm for reaching the optimal services combination. The objective function to be solved must be linear and cannot be used to completely solve the Web service composition problem (Gao, 2021; Kurokawa, 2021; Liu et al., 2021).
(3) The swarm intelligent optimization algorithm overtakes the shortcoming of the global optimization algorithm. It is a popular method currently used for Web service composition (Abualigah et al., 2021, 2022; Liu et al., 2019; Kashyap et al., 2020).

However, These swarm intelligence algorithms have the problem of trapping at local optima to some extent. To face the massive requirements and available services, the performance of the aforementioned service composition methods cannot satisfy the needs of users. Therefore, it is necessary to improve the swarm intelligence algorithms according to the features of the service composition problems, in order to make it perform better in the service compositions. On the other hand, efficiencies of the service composing could be improved once the huge number of existing service history records is used as a prior knowledge to guide the service composing.

This paper introduces a productive and effective method for service compositions. The proposed approach uses domain knowledge to mine the relationships and fixed collocations between services in order to build large-grained service units (service patterns). Since the service patterns are large-grained, the search space is reduced. The service patterns are fixed collocations, which are verified by the historical service composition solutions. Hence, the users' satisfactions are the higher. Meanwhile, the improved particle swarm optimization algorithm composite services for the partial requirements those are not covered by the service patterns. It can effectively overcome the dilemmas of easy trapping at local optima, such that the efficiencies of compositions can be improved. In this work, the experiment results on the public service datasets demonstrate that the proposed method has high performance on service composition.

The rest of this study is organized as follows. "Related work" section describes the introduces swarm intelligence algorithms of service compositions. "Web service composition problem descriptions" section describes the service composition problem and the construction of service patterns. "Proposed IDPSO algorithm" section explains the proposed method of improved discrete particle swarm optimization (IDPSO) and improved discrete particle swarm optimization based on prior knowledge (PK-IDPSO) algorithms. In "Numerical experiments" section, the superiority of the proposed method is evaluated by comparing the performance with the state-of-the-art algorithms.

Related work

The QoS-SC problem has three folds: optimality, convergence and time complexity. Time complexity indicates the time-consuming of the algorithm; the convergence indicates the speed of the algorithm's optimization while the optimality reveals its optimization ability. Generally, the time complexity and the optimality are contradictory, which means that seeking high time efficiency will sacrifice the optimality of the algorithm, and vice versa. Therefore, reaching a balance between those two metrics is the key to the customer needs in the service composition. In addition, improving the convergence of the algorithm is also important to customer needs. The challenge is to manipulate the corresponding specific services from the candidate set to meet the QoS restrictions proposed by customers, which is an NP-hard problem (Mabrouk, 2009). The current swarm optimization algorithm is effective for continuous problems, but it is limited for discrete problems such as web service composition. All swarm optimization algorithms have encountered the local optimal issue because that they follow a certain direction and order during the optimization. Jin et al. (2015) proposed a service pattern that uses the genetic algorithms for service composition solutions by selecting a service composition with association awareness in cloud manufacturing. Wen et al. (2013) used the Particle Swarm Optimization (PSO) algorithm to solve the Web service composition optimization problem. The PSO algorithm compared with the genetic algorithm, it benefits from its fewer parameters and fast convergence, and showed better performance for the problems such as Optimizations. However, it also traps in the local optimal problem (Chen, 2014). Zhang et al. (2015) proposed a method to classify candidate services for large-scale Web service composition.This method proposes a reduction model to reduce the size of the service candidate set, in an attempt to reduce the search space of the problem to improve the optimization efficiency. This method needs to reduce the new candidate service set according to the new service composition problem every time, so that reduce the overall efficiency of the algorithm. Liu et al. (2019) proposed the Artificial Bee Colony (ABC) algorithm. The

Artificial bee colony algorithm is an algorithm constructed to simulate the behavior of bees in the process of collecting honey. Compared with the PSO algorithm, the ABC algorithm has fewer parameters, which gives better performance regarding time complexity, but its performance in optimality is slightly lower. Zhang et al. (2018) proposed a task-granularized quality-constrained-aware service composition method. They divide the service composition model into four layers. The basic layer is related to the candidate service set of the service composition where the top layer is related to the task. The granular layer corresponds to the tasks that are segmented in the service composition. The layered task granulation method can improve the composition efficiency, but it breaks the overall constraints of the service composition plan, such that the fact that the service composition plan found may not be optimal. Zhang et al. (2019) introduced an extended flower pollination algorithm (FPA) for solving QoS-SC problems. Zhang et al. (2021) proposed a novel approach to construct quality relevance index graph to realize efficient query of quality relevance. Jatoth and Gangadharan (2019) proposed a modified invasive weed optimization (IWO) algorithm to obtain the optimal solution. Khanouche et al. (2019) presented an approach based on the k-means clustering technique to group candidate services into clusters and a search tree to find the near-optimal solution. Based on service dependency graph, Zhang et al. (2021) introduced a top-k service composition approach to find suitable services in IoT environments. Chen et al. (2016) considered the QoS difference compared with the customer's requirement (QoS risk) as an individual objective besides either each QoS attribute and introduced an optimization method named Efficient-Dominance Multi-Objective Evolutionary Algorithm (EDMOEA) to solve QoS-SC problems. Zhou and Yao (2017) presented energy consumption formulation formulas for software/hardware cloud services. The global QoS value and energy consumption are considered as two objectives to be optimized. Then a Multi-Objective Hybrid Artificial Bee Colony (MOHABC) method is proposed to generate the Pareto optimal solutions. Huo et al. (2017) presented a novel multi-objective service composition model that takes the global QoS value and the cost of composite services as two objectives for QoS-SC problems. Wang et al. (2022) proposed a service composition exception handling adaptive adjustment (SCEHAA) algorithm based on the improved ant colony optimization algorithm (ACO) and applied to address QoS-SC problems. Liang et al. (2021) are dedicated to exploring possible applications of deep reinforcement learning(DRL) in QoS-SC and a logistics-involved QoS-aware DRL-based QoS-SC method is proposed. Seghir and Khababa (2018) proposed a hybrid genetic algorithm (HGA) to solve QoS-SC problems. This algorithm combines two phases to perform the evolutionary process search, including genetic algorithm phase and fruit fly optimization phase. Li et al. (2020) proposed a SDF(service domain features)-oriented genetic algorithm to effectively create a manufacturing service composition with large-scale candidate services. Li et al. (2021) based on the core principle of evolutionary methods, first developed an elite evolutionary strategy (EES) and then utilized it to advance convergence speed and ability of Harris hawks optimization(HHO) to jump out of the local optimum. Sangaiah et al. (2020) proposed an efficient method for solving the QoS-SC problem using biogeography-based optimization (BBO). BBO is a very simple algorithm with few control parameters and effective exploit. Sefati and Navimipour (2021) proposed an effective way based on a hidden Markov model (HMM) and an ant colony optimization (ACO) to solve the QoS-SC problem by enhancing the QoS.

The above service composition algorithms do not focus on balancing the global and local search abilities in the process of algorithms optimization, and they do not consider reducing the search space with prior knowledge. This paper proposes a Web service composition method based on prior knowledge and an IDPSO algorithm. A set of service patterns has been identified from a large number of historical service records. The service patterns are fragments of service processes that frequently appear in historical service plans. Proper use of the service patterns could make a considerable part of the service demand process without immediately selecting unit services. Only service patterns matching are required. Moreover, for the fragments of the service demand process that did not match successfully, the IPSO algorithm is performed instantaneously to apply local service combinations. A mechanism is introduced by the IPSO algorithm to avoid the local optimal. Both theoretical analysis and experimental results show that the proposed method achieves better performance in terms of time complexity and optimality for web service composition optimization.

Web service composition problem descriptions

QoS

The QoS is described by:

$$QoS = (p, r, a, rep, t, rel) \tag{1}$$

where p is the service price, r is the response time, a is the availability that refers to the ratio of the number of successful executions out of total number of executions, t is throughput, and rel is reliability that refers to the ratio of the time the service can run to the total time, rep is reputation which is calculated by:

$$rep = \left(\sum_{i=1}^{n} sc_i\right)/n \tag{2}$$

Where rep is the average value of the user's n evaluation scores ($sc_1, sc_2, ..., sc_i, ..., sc_n$) of the current service.

Service composition problem

The purpose of the service composition is to obtain a set of optimal services to form a composite service for a specific purpose under the satisfaction of users' requirements and overall QoS constraints. The composite service can be expressed by:

$$CS = (T, R, QoS, QC, W, EF) \tag{3}$$

where $T = \{t_1, t_2, ..., t_j, ..., t_n\}$ is the set of tasks that constitute the composite service, the relation $R \subset T * T$ represents the set of timing relations between tasks, QoS is the overall quality of the composition service. $QC = C(p), C(r), C(a), C(re), C(t), C(rel)$ represents the quality constraints that must be met by the composite service,$w = \{w_1, w_2, ..., w_6\}$ is the weight corresponding to each subattribute in the set QoS s.t. $\sum_{i=1}^{6} w_i = 1$. EF is the composition service quality evaluation function, i.e., the optimality function.

Task t_j in task set T is an abstract service, a specific web service as_k^j is selected from the candidate service set CSS_j of t_j to replace the abstract service CSS_j, where as_k^j represents the $k-th$ candidate service in CSS_j,$k \in \{1, 2, ..., \|CSS_j\|\}$. The generation process of the composite service is to find the composite service process that meets the customer's needs. The process of generating composite services is mainly divided into two steps, service selection and service composition. In the service selection process, the task T submitted by the customer is abstracted into multiple subtasks$\{t_1, t_2, ..., t_j, ..., t_n\}$, and each subtask has a different execution relationship. The Workflow Management Coalition (WFMC) defines four basic execution relation models, Sequence, Parallel, Selective and Circular. The Parallel, Selective and Cycle models can be transformed into Sequential models. The sequential models are selected in this work. Since customers have specific requirements for QoS attributes, according to the overall QoS value of the combined service, the combined evaluation function EF is used to evaluate the solution of the combined service and the best combination is selected. Since the value range of each attribute of QoS of Web service is different, the QoS attribute value is normalized (Wen et al., 2013). The smaller the values of each sub-attribute of QoS, the better the service.

Construction of service pattern

The FP-Growth (Liu, 2017) is employed to construct service patterns (Xu, 2020). For new customer task processes, the existing service patterns are first matched. Then, the optimization algorithms are used to perform partial tasks for unmatched task process fragments. Instead of combining all the optimization algorithms of the task process at the beginning, the introduction of service pattern can greatly improve the efficiency of service composition. The service pattern can be represented by:

$$SP = (ID, PaSeq, PaFre) \tag{4}$$

where ID is the identity number of the service pattern, and uniquely represents the specified service pattern:

$$\begin{aligned} PaSeq \ = \{ &< as_1, as_2 >, < as_2, as_3 >, ..., < as_i, \\ &as_{i+1} >, ..., < as_{n-1}, as_n > \} \end{aligned} \tag{5}$$

There is a sequence relationship between as_i and as_{i+1}, whose order cannot be reversed. $PaFre$ represents the frequency of service patterns, as shown in Fig. 1

Demand matching based on service pattern

Two indicators, $Coverage$ and $Redundancy$, are used to evaluate the coverage and efficiency of the service pattern. The Coverage $cov(T, SP_i)$ is the ratio of the number of unit tasks in the service pattern SP_ithat can cover the task T,$\|C_{sp_i}\|$, and the total number of unit tasks in the task process proposed by the customer, C_T, i.e. $cov(T, SP_i) = \|C_{sp_i}\|/C_T, cov(T, SP_i) \in [0, 1]$. The Coverage reflects the extent to which a service pattern fits the customers tasks. The Redundancy $redu(T, SP_i)$ is the ratio of the number of unit services that have not been successfully covered in the service pattern, $\|A_{SP_i}\| - \|C_{SP_i}\|$ to the total number of unit services in the service pattern,$\|A_{SP_i}\|$, i.e. $redu(T, SP_i) = \|A_{SP_i}\| - \|C_{SP_i}\|/\|A_{SP_i}\|, redu(T, SP_i) \in [0, 1]$. In the actual service composition process, some unit services in the service pattern do not cover the unit services in the task flow.

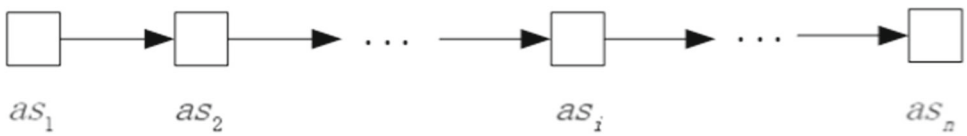

Fig. 1 Schematic diagram of PaSeq

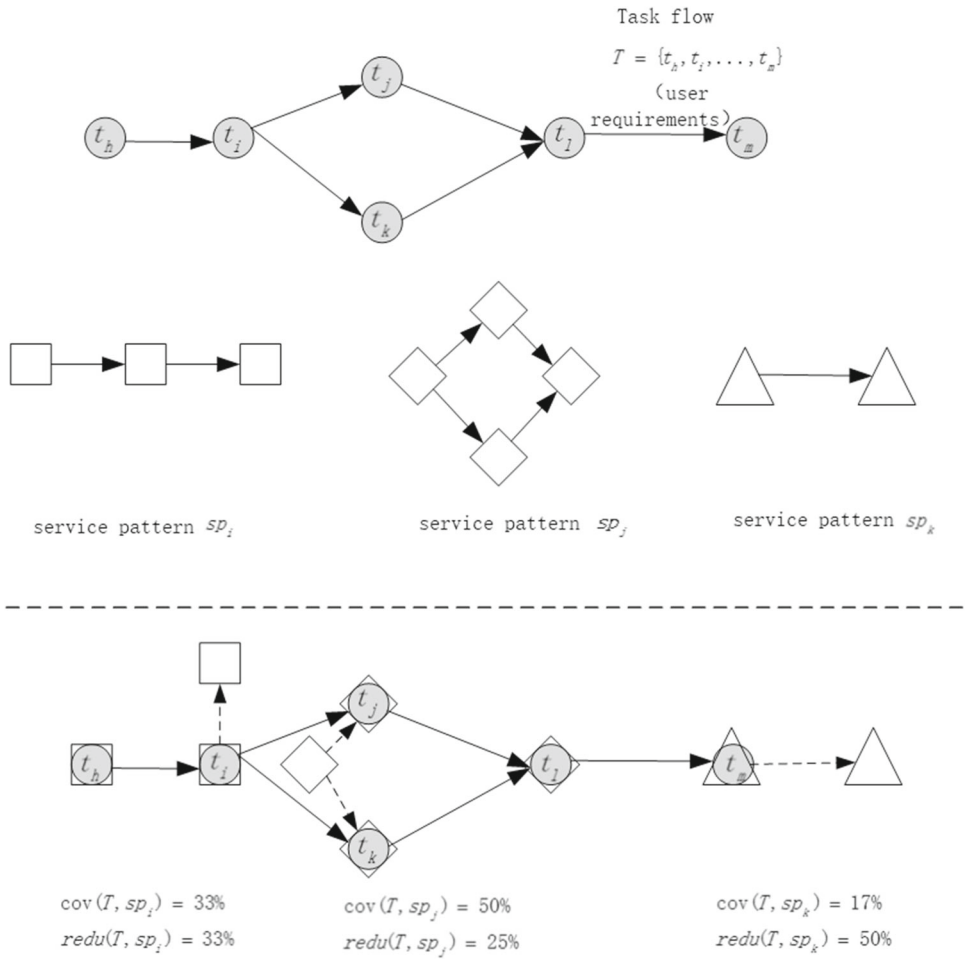

Fig. 2 Example of coverage and redundancy of the service patterns

The Redundancy can be used to measure the degree of redundancy of the service pattern. The higher the redundancy, the more invalid unit services and higher cost. Fig. 2 shows an example of coverage and redundancy.

The first step of the proposed PK-IDPSO algorithm is to obtain a set of matching tasks based on the service patterns, using the mapping algorithm SP-Mapping to target the task flow in the customer's requirements. The SP-Mapping algorithm is based on the KMP algorithm (Hongwei et al., 2006), which is a string pattern matching algorithm. Pattern matching of strings is a common operation. The so-called pattern matching can be simply understood as looking for a given pattern (string) in the target (string), returning the first character position of the first substring matched by the target and the pattern. Usually, the target string is relatively large while the pattern string is relatively short. The KMP algorithm is an improved string-matching algorithm, which uses the information after the matching failure to minimize the matching times between the pattern string and the main string to achieve the purpose of fast matching. The specific implementation is through a $next()$ function, which itself contains the local matching information of the pattern string. The time complexity of the KMP algorithm is O(m+n), where m is the length of the main string and n is the length of the substring (Hongwei et al., 2006).

The SP-Mapping algorithm selects the corresponding service patterns to cover the task fragments in the task process. Generally, a service pattern covers some tasks in the task process. A task process always expects more tasks to be covered by a service pattern. The fine-grained matching is adopted, i.e., all unit services in a service pattern need to cover the tasks in the task process such that the redundancy is zero, $redu(T, SP_i) = 0$. In order to make the coverage rate of the specified service pattern $cov(T, SP_i)$ greater, the SP-Mapping algorithm first sorts the service patterns in descending order according to their lengths, so as to match the service patterns with a larger coverage rate. Fig. 3 shows an example of service pattern mapping.

After the task flow $T = t_o, t_p, ..., t_z$ is matched by the service pattern mapping algorithm, the task flow fragment t_o, t_p, t_q is covered by the service pattern sp_α, the task flow fragment t_r, t_u, t_v is covered by the service pattern sp_β, the task flow fragment t_w, t_x is covered by the service pattern sp_η. The task flow fragments t_s, t_t and t_y, t_z are not successfully

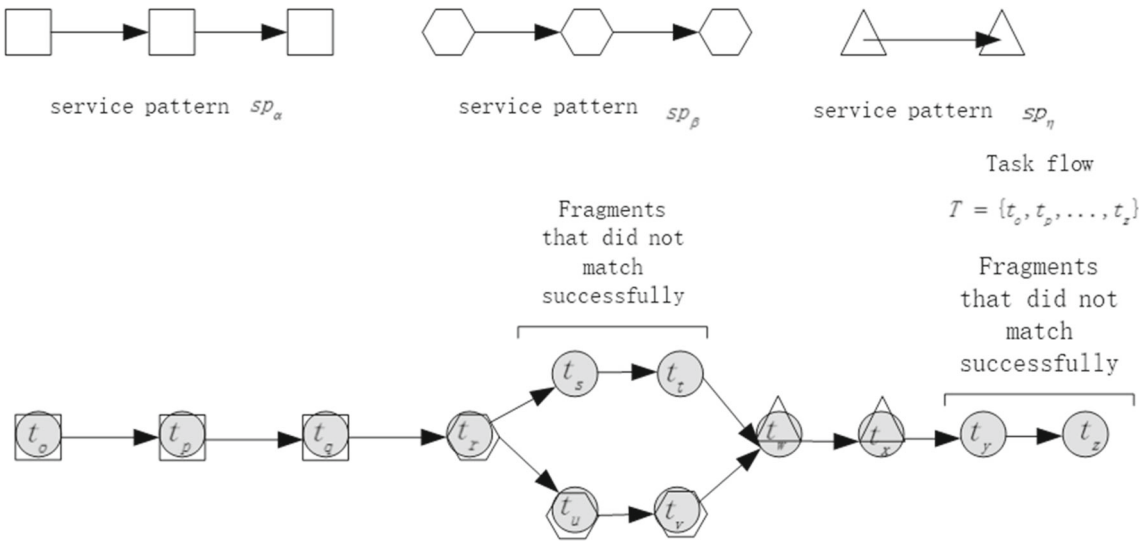

Fig. 3 Service pattern mapping

mapped by any service pattern, and the PSO needs to be called for partial service composition.

Algorithm 1 The service pattern mapping algorithm

Input:
1: T,SP
Output:
2: $MapMark$
3: descending sort to SP
4: Initialize the length of $MapMark$
5: Initialize $MainString$
6: **while** $MainString \neq 0$ **do**
7: **while** SP_i mactching is not success and is not the last one **do**
8: calling kmp algorithm to match $MainString$ with SP_i
9: updating SP_i
10: **end while**
11: updating $MapMark$
12: updating $MainString$
13: updating updating SP_i
14: **end while**
15: return $MapMark$

- T is the task process given by the customer.
- SP is the candidate service pattern set.
- $MapMark$ is the mapping mark table used to record the situation covered by SP in T, where M$apMark[i] = None$ represents no match, $MapMark[i]$ is equal to the ID of the service pattern No. indicates that the matching is successful. The service pattern that is matched is the service pattern corresponding to the ID number, $MapMark[i] = -1$ indicates that it has been matched but has not been matched successfully.

Proposed IDPSO algorithm

Standard particle swarm optimization algorithm

PSO algorithm is a heuristic evolutionary computing technology (Wen et al., 2013; Eberhart, 1995) derived from the simulation of simplified social group intelligent behavior model. At the t+1 generation, the iterative update formula for the j-th dimension velocity and position of the i-th particle is expressed by:

$$v_{i,j}^{t+1} = w * v_{i,j}^{t} + c_1 r_1 (p_{i,j} - x_{i,j}^{t}) + c_2 r_2 (p_{g,j} - x_{i,j}^{t}) \quad (6)$$

$$x_{i,j}^{t+1} = x_{i,}^{t} + v_{i,j}^{t+1} \quad (7)$$

where the non-negative constants c_1 and c_2 are learning factors, which determine the degree of influence of $p_{i,j}$ and $p_{g,j}$ on the new speed, and r_1 and r_2 are uniformly distributed random variables within [0, 1]. The linearly decreasing inertia weight value proposed by Eberhart and Kennedy (1995) is widely used:

$$w = w_{min} + (w_{max} - w_{min})(T_{max} - t)/T_{max} \quad (8)$$

where w_{max} is the initial inertia weight value, w_{min} is the inertia weight value corresponding to the maximum iteration, w_{max}, w_{min}.In the service selection of service composition, a particle is regarded as a service composition solution. The abstract subtask $T = t_1, t_2, .., t_i, ..., t_n$ decomposed represents the search space of each dimension of the particle, n represents the number of abstract subtasks decomposed, it also indicates that the search space of the particle has n dimensions. The number of web services in the candidate service set css_i corresponding to each abstract subtask t_i is

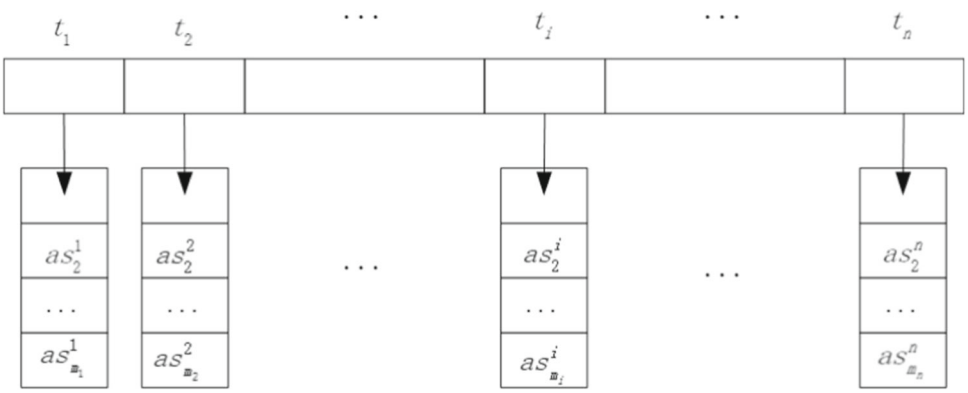

Fig. 4 Particle coding in service composition

set to m_i, which means the maximum vector length of the particle in this dimensional search space, $i \in [1, n] \wedge i \in N^+$. Each element in css_i is the unit service as_j^i, which is specifically coded, as shown in Fig. 4.

After the particles select the corresponding vector values of each dimension, the selection of the web service composition solution is completed. The web service composition solution corresponding to each particle is evaluated by the fitness value evaluation function to find the optimal service composition solution. Using this coding method, the web service composition selection problem is transformed into an n-dimensional vector solution problem, that is, the process of finding the optimal vector $(x_1, x_2, ..., x_i..., x_n)$. Task $T = t_1, t_2, ..., t_i, ..., t_n$ is the task set of the composite service CS. The number of candidates corresponding to each abstract service in T is $m_1, m_2, ..., m_i, ..., m_n$ and $m_i \in N^+ \wedge i \in [1, n]$. Then the size of the composite service solution space is $\prod_{i=1}^{n} m_i$, which is also the time complexity of using the exhaustive method to find the optimal solution. When the number of subtasks in task T and the value of the candidate service m_i corresponding to each subtask t_i are very large, the time complexity is very large, which is the direct reason why the service composition is not suitable for exhaustive methods. The IDPSO algorithm sets the movement intervals of particles in all its dimensions as continuous circular orbits. When the fitness function is used to evaluate the pros and cons of particle positions, the divisor function is used to map the specific positions of the particles to discrete intervals.

Improved discrete particle swarm optimization algorithm

Fitness function

As been discussed in the previous section, the overall QoS of the service composition solution has a greater impact on the service evaluation. According to the QoS attribute value of a single node service in the service composition solution, aggregate calculations are performed to obtain the overall QoS index of the service composition solution.

$$Q_{cs}^r = Aggregation(cs_r) \tag{9}$$

Where cs_r is the rth attribute of the current service composition solution, $Aggregation(cs_r)$ is the aggregation of the rth attribute of each atomic service in the service composition solution cs. The fitness function is used as the evaluation function of the Web service composition solution.

$$fitness = \sum_{r=1}^{n} w_r * Q_{cs}^r \tag{10}$$

where w_r represents the customer's preference for the rth QoS attribute of the Web service composition solution, n is the total number of QoS attributes of the service composition solution. Smaller the fitness value indicates better optimality.

Premature prevention mechanism

The premature particle swarm means that the entire particle swarm algorithm has fallen into a local optimal, or the fitness value has not been updated for a long time after finding a local optimum. This long-time duration could not be accepted by customers. To this end, an escape mechanism is thus built:

Candidate service set sorting It is found that in the PSO algorithm, each particle will first randomly initialize a position, and then move in a specific direction at various orbits. Therefore, the particle swarm algorithm is suitable for the problem whose optimal solution has a certain direction and order. As shown in Fig. 5, the particle swarm algorithm is used to find the position of minimum value for the parabolic function $y = f(x)$. The position of a particle is currently at the position of x_0. The particle can approach the position of the lowest point X_1 only by moving to the right. For this kind of orderly problem, the direction of at least one particle in the particle swarm is exactly the same as the direction of the solution. At this time, the optimization problem is solved.

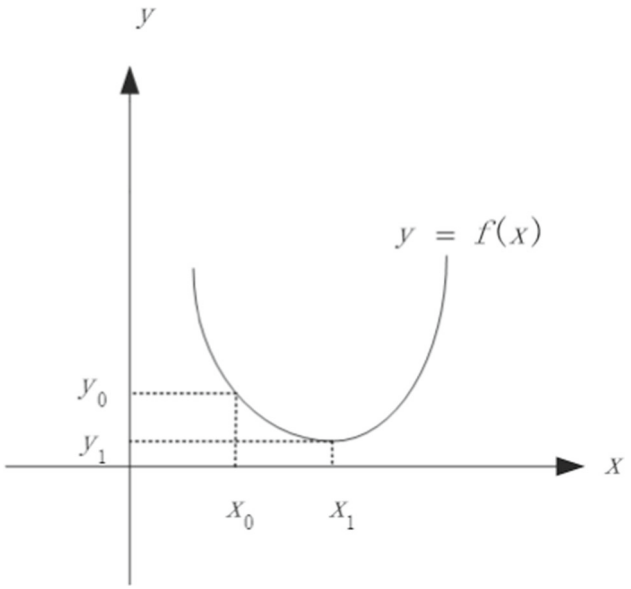

Fig. 5 Illustration of finding the optimal solution

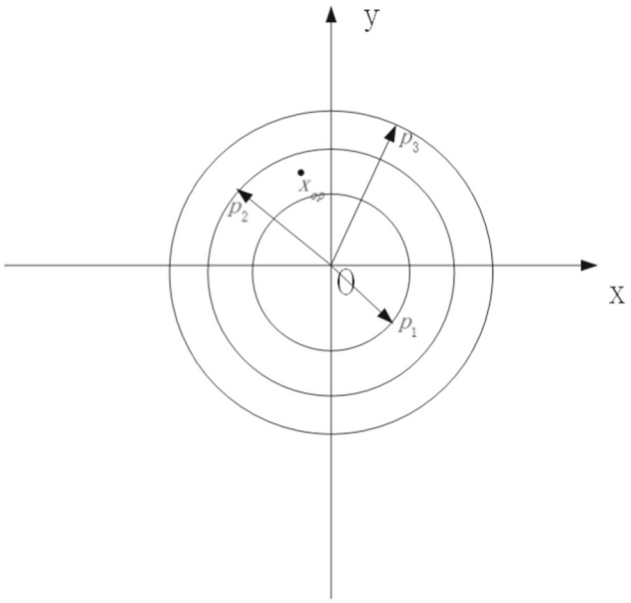

Fig. 6 Illustration of premature

It is found that the manipulation of the candidate service set by the IDPSO algorithm is ultimately reflected in each QoS attribute of each atomic service as_j^i in the CSS_i. Therefore, the CSS_i are sorted according to the QoS attribute with the largest weight. The weight corresponding to the six attributes in $QoS = (p, r, a, re, t, rel)$ is $w = (w_1, w_2, w_3, w_4, w_5, w_6)$. The first two attributes of QoS are reverse attributes, that is, the larger the attribute value, the worse the satisfaction, the latter four attributes are forward attributes, larger attribute value, indicates better satisfaction.

Particle position resetting When the particle swarm algorithm starts, it will first randomly generate a position for each particle, and then each particle will move on its orbit to find the optimal position. The current trajectories of all particles happen to miss the global optimal or better position, then the entire particle swarm algorithm falls into the local optimal, as shown in Fig. 6.

This particle group consists of three particles, p_1, p_2 and p_3. x_{op} is the optimal position or better position in the entire space, x_{op} is between the orbit of particle p_1 and the orbit of particle p_2, the three particles p_1, p_2, and p_3 are moving on their respective orbits. The orbits of the three particles never cross x_{op}, which means that the particle has fallen into a local optimal, and no matter how long thereafter, the particle will not find the x_{op} of the global optimum or a better position. Therefore, two steps need to be taken for particles escaping from the local optimal position. Firstly, when the particle swarm algorithm has not changed for a long time, its optimality value does not change, and it is deemed as falling into the local optimum. The time is set to t_{pre}, s.t. $t_{pre} \in [\frac{1}{5}T_{max}, \frac{1}{3}T_{max}] \wedge t_{pre} \in N^+$ .Secondly, when the particle swarm algorithm falls into the local optimum, we use the random reset method to reset the position of each particle so that the trajectory of one or some particles exactly swipes the optimal position or better position as much as possible.

Control parameter setting The control parameters of IDPSO are PS, T_{max}, c_1, c_2, w, v_{ini}. PS represents the size of the particle swarm (the number of particles), T_{max} is the maximum number of iterations, c_1, c_2 are non-negative constants, which determines the influence of the optimal position of the global particle and the optimal position of the current particle history on the particle velocity, w is the inertia weight, and v_{ini} is the initial velocity of the particle. The parameters PS and T_{max} directly affect the time complexity and the optimality of the algorithm, because the larger the $PS \times T_{max}$, the wider the search range of this algorithm, the larger the value of $PS \times T_{max}$, the better the optimality. Meanwhile, it will increase the time complexity of the algorithm. Therefore, it is necessary to choose the value of $PS \times T_{max}$ to ensure the optimality of the algorithm within the time that the customer can tolerate. The solution space of all the service composition solutions of the IDPSO algorithm is $\prod_{i=1}^{n} m_i$, when $PS \times T_{max} = \prod_{i=1}^{n} m_i$, the IDPSO algorithm exhausts the entire solution space, the global optimal solution can be found. When $PS \times T_{max}$ is very large, $PS \times T_{max}$ cannot be equal to $\prod_{i=1}^{n} m_i$. We generate a large number of PS and T_{max} parameters through the generation tool, and then input these parameters into the IDPSO algorithm, and finally get the experimental result data, and then

we analyze the experimental data, and the results show that when the value of $PS \times T_{max}$ is at least $\frac{1}{1000}\prod_{i=1}^{n} m_i$, the IDPSO algorithm can achieve good optimality, thus we set $PS \times T_{max} = \frac{1}{1000}\prod_{i=1}^{n} m_i$. The IDPSO algorithm only needs to exhaust about one thousandth of the entire solution space. A satisfying optimality can be achieved. When the value of $PS \times T_{max} = \frac{1}{1000}\prod_{i=1}^{n} m_i$ is still intolerable, it can be adjusted according to the customer's needs. We assign the values of PS and T_{max} in a one-to-one manner. c_1 and c_2 are learning factors that determine the impact of the historical global optimal position and the current particle historical optimal position on the new velocity. If the values of c_1 and c_2 are larger, the historical global optimal position of the particle swarm and the current historical optimal position of the particle will have a greater impact on the new velocity, and vice versa. There are no standard values for c_1 and c_2 yet. The suggestion is to first assign large values for c_1 and c_2. Once the optimality value of the IDPSO algorithm changes frequently, it means that the historical global optimal position and the current particle historical optimal position are actually very helpful for the algorithm to search for the new optimality value. Therefore, at this time, we need Maintain or increase the influence of the historical global optimal position and the current particle historical optimal position on the algorithm optimization, so we can choose not to change or appropriately increase the values of c_1 and c_2. Once the optimality value of the IDPSO algorithm has not changed after many generations, it means that the historical global optimal position and the current historical optimal position of the particle are actually not very helpful or helpful for the algorithm to search for the new optimality value. Therefore, At this time, we need to reduce the influence of the historical global optimal position and the current particle historical optimal position on the algorithm optimization, so we can appropriately reduce the values of c_1 and c_2. The parameter w can be set according to Eq. 8. The parameter v_{ini} is the initial velocity of the particles. Since the global optimal position is not accessible, a random strategy is applied to set a value for the parameter v_{ini}, the random strategy is to randomly generate a set of integers as the value of v_{ini}.

Reference setting In the relationship between the whole and the components, we know that the optimal components does not mean the optimal overall, but the combination of the optimal components is better, so the optimal components has a certain reference value for the overall optimization. We apply this concept to the IDPSO algorithm, task $T = t_1, t_2, ..., t_i, ..., t_n$ earlier, and its corresponding candidate service set is $CSS = css_1, css_2, ..., css_i, ..., css_n$, that is, the candidate service set corresponding to subtask t_i is ccs_i. First, we adapt the unit service as_j^i computing unit in the candidate service set ccs_i of each subtask t_i:

$$fitness_{as} = \sum_{r=1}^{n} w_r * Q_{as}^r \tag{11}$$

where w_r is the weight of the rth attribute of the unit service as, Q_{as}^r is the rth attribute of the unit service as, and then the unit service as_{MF} with the smallest unit fitness value in each candidate service set css_i is selected to form the reference optimal service composition solution cs_{ref}, and calculate the fitness value of the solution with formulas 9 and 10 , which we call the reference fitness value $fitness_{ref}$. When the IDPSO algorithm executes the last iteration, compare the fitness value $fitness$ of the obtained global optimal particle with the reference fitness value $fitness_{ref}$. If $fitness < fitness_{ref}$, it indicates that the particle swarm algorithm has selected the global optimal or better solution. At this time, the solution corresponding to the global optimal particle is used. If $fitness > fitness_{ref}$, it indicates that the particle swarm algorithm is actually trapped in the local optimum. At this time, if the customer cannot accept the long wait caused by the particle swarm optimization continuing to optimize, the customer should choose to refer to the optimal service composition solution cs_{ref} as the optimal solution for the service composition. If $fitness = fitness_{ref}$, it indicates the optimality of the solution selected by the particle swarm algorithm comparing with the optimality of the reference optimal service composition solution cs_{ref}, the particle swarm algorithm may still fall into the local optimum. if the customer cannot accept the long wait caused by the particle swarm optimization continuing to optimize, then it does not matter whether the optimal service composition solution selected by the current particle swarm is used or the reference optimal service composition solution cs_{ref} is used.

- PS is the number of particles.
- X is the positions matrix of the particles.
- P_{ibest} is the individual historical optimal positions of the particles.
- P_{gbest} is the global historical optimal positions of the particles.
- $fitness_{ref}$ is the fitness value of the reference optimal solution.
- $fitness_g$ is the fitness value of the global optimal solution given by the loop statement, and the IDPSO algorithm finally outputs the position of the optimal particle.

Framework of the proposed approach

The service patterns are a set of service process fragments excavated from a large number of historical service records by using data mining techniques. The service pattern, the matching algorithm SP-Mapping, and the IDPSO algorithm are integrated to carry out an overall service composition for

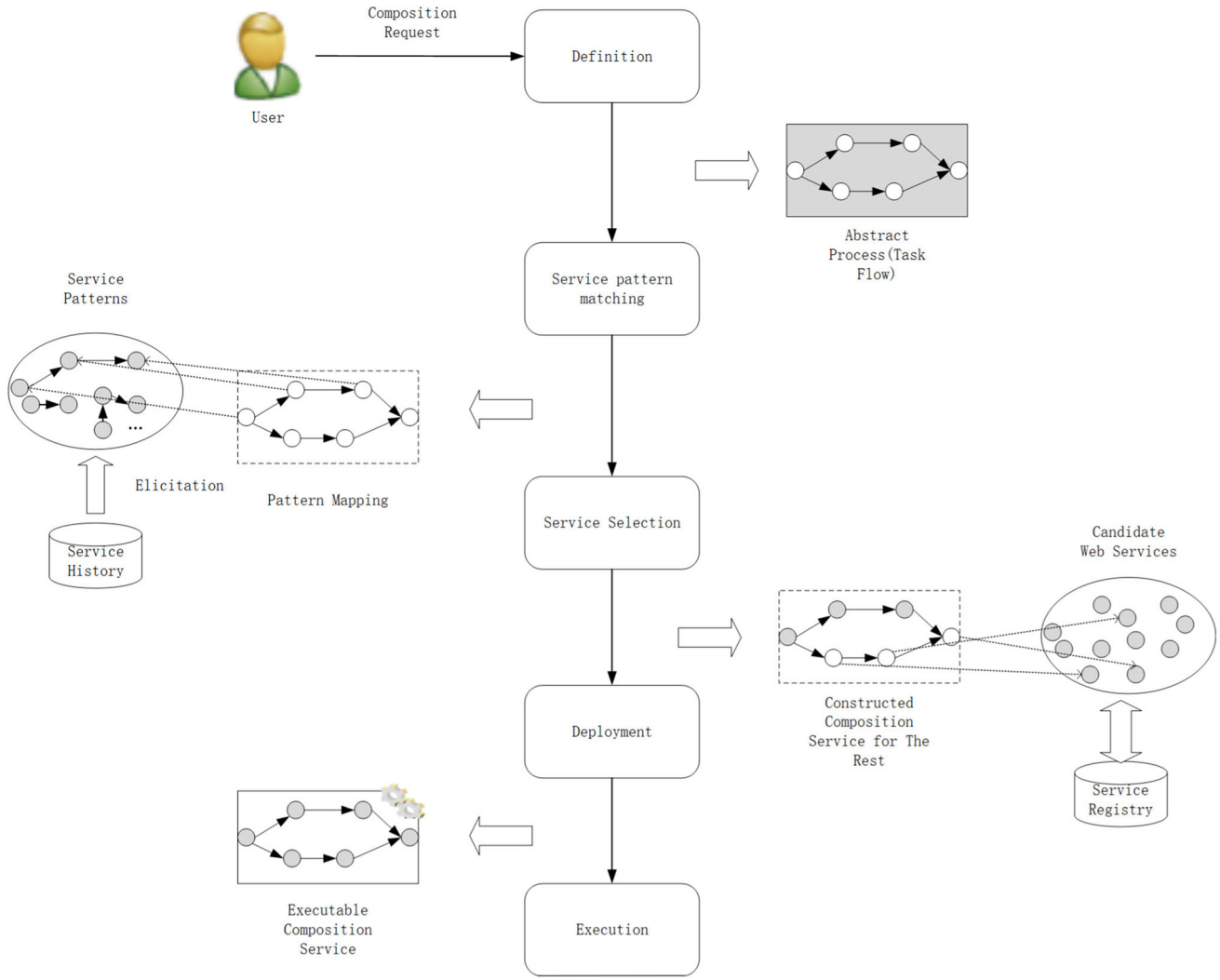

Fig. 7 Proposed PK-IDPSO algorithm framework

the task flow of the demanding service. Firstly, SP-Mapping is applied to map the process of the new tasks. Then, the IDPSO algorithm is used for the part of the task process T that has not been successfully mapped to perform local dynamic and instant service composition. The proposed PK-IDPSO algorithm framework is shown in Fig. 7. In Fig. 7, the user submits a service composition request to the model. The model processes the request submitted by the user into a unified abstract task flow and pushes it to the service pattern matching phase. In the service pattern matching phase, the model uses the pre-mined service patterns and the proposed SP-Mapping algorithm matches the part of the current task flow, and submits the unmatched remaining part to the service selection phase. In the service selection phase, the model uses the proposed IDPSO algorithm proposed to composite services for the remaining part of the current task flow, and finally the model integrates the results into a complete service composition solution and feeds it back to the user.

Numerical experiments

Experimental settings

In this work, the QWS public data set of Zeng et al. (2003, 2004) is adopted for the atomic service part. We test the number of different subtasks separately, and assign the services on the QWS data set to these subtasks accordingly, which are used as the candidate service set of the subtasks. The candidate set of each subtask includes 100 units. Four attributes are selected and extra two attributes are supplemented by simulation that is to randomly generate values according to the value range of the corresponding attribute values and

Table 1 System resource configuration

OS	CPU	RAM	Experimental tool
Windows 7 64-bit	Intel(R) Core(TM) i5 3230 3.60GHZ	12GB	eclipse+pyDev2.7+python 3.0

Table 2 Service pattern data set

Service pattern attributes	Number of service patterns	Length of Service patterns
ID, service sequences	5154	2-20

Table 3 Atomic service data set

Total	QoS attributes and value range
2000	$r(1-5000), a(1-100), t(1-10), rel(1-100), p(1-100), rep(1-10)$

Algorithm 2 IDPSO algorithm

Input:
$PS, T_{max}, c_1, c_2, w, V$
Output:
P_{gbest}
Initialize PS, c_1,c_2
Set dimensions of particles
Descending or ascending sort for candidate service set css by QoS attribute of max weight value
Randomly initialize V, X
Compute fitness
Set P_{ibest},P_{gbest} according to fitness
Set $t_{pre} = 1$
while $iterations <= T_{max}$ **do**
  if t_{pre}=set count **then**
    Randomly initialize V, X
  else
    Update c_1, c_2, w
    Update V
    Update X
  end if
  Compute fitness
  Update P_{ibest},P_{gbest} according to fitness
  t_{pre}++
end while
Compute $fitness_{ref}$
if $itness_{ref} < fitness_g$ **then**
  return P_{ref}
else
  return P_{gbest}
end if

insert the values into the atomic service data set. A total of six attributes are used as the QoS evaluation index attributes, response time (r), availability (a), throughput (t), Reliability (rel), service price (p), and reputation (rep). The first four are the original attributes in the QWS data, the last two are the attributes inserted by simulation. The weight is set as $weight(r, a, t, rel, p, rep) = (0.2, 0.3, 0.1, 0.1, 0.2, 0.1)$ to the customer preferences. In addition, we collected 10,000 simulated historical service records, and mines 5154 service patterns from them as the data set of the service pattern matching algorithm SP-Mapping.The system resource configuration and data set characteristics of this experiment are listed in Tables 1, 2, 3.

We set up some groups of experiments according to the number of different sub-tasks. The number of sub-tasks in the first group is 2, and the number of sub-tasks in other groups is increased by 2. The number of sub-tasks in the last group is 20, experiments of each group are performed 50 times, and the final result was the average of 50 times. The parameter settings of experiments of each group are listed in Table 4. The values of $c_1$and c_2 are dynamically set by the algorithm according to "" section 4.1. The inertia weight w is dynamically changed according to Eq. 8 and the initial values are given according to Eq. 9. The initial values of the particle velocity V and the particle position X are generated randomly. Those values will be dynamically changed according to Eqs. 6 and 7 .

Results and discussion

The IDPSO, PK-IDPSO, ZP-PSO (Kashyap et al., 2020), S-ABC (Liu et al., 2019), EO (Jin et al., 2022), RSA (Abualigah et al., 2022), GA (Cergibozan & Tasan, 2022), FPA (Zhang et al., 2019), HAA-ACO (Wang et al., 2022), HHO (Li et al., 2021), BBO (Sangaiah et al., 2020) and HMM-ACO (Sefati & Navimipour, 2021) algorithms are compared by evaluating the performances in terms of Optimality, Convergence and Time Complexity. Optimality reflects the quality of the solution obtained by the algorithm. The better optimality means that solution obtained by the algorithm can meet the user's requirements is better. Convergence reflects the speed of algorithm optimization. The better convergence of the algorithm means that optimization speed of the algorithm is faster. Time complexity refers to the time it takes for an algorithm to complete a service composition. So, if the algorithm time complexity is smaller then the algorithm is better. Obviously, the optimality and time complexity of an algorithm are contradictory. The number of iterations and

Table 4 Parameters for different numbers of subtasks

TasksNum	Params							
	PS	T	t_{pre}	c_1	c_2	w	V	X
2	4	4	2	100	100	0.7750	ram	ram
4	50	50	10	100	100	0.8990	ram	ram
6	100	100	20	100	100	0.8950	ram	ram
8	250	250	50	100	100	0.8980	ram	ram
10	500	500	100	100	100	0.8990	ram	ram
12	700	700	140	100	100	0.8993	ram	ram
14	900	900	180	100	100	0.8994	ram	ram
16	1000	1000	200	100	100	0.8995	ram	ram
18	1500	1500	300	100	100	0.8997	ram	ram
20	2000	2000	400	100	100	0.8998	ram	ram

the number of individuals for the same number of subtasks are the same for each compared algorithm.

Optimality

The Optimality is expressed by:

$$Optimality = fitness \tag{12}$$

The smaller the Optimality value, the higher its optimality. In order to verify the optimality of the algorithm proposed for web service composition selection, the number of candidate unit services selected for each subtask is set to 100, and the ability of the five algorithms to find the optimal solution under different subtasks is investigated. The results are listed in Table 5 and Fig. 8.

It can be seen from Table 5 and Figure 8 that in the different subtask tested, except PK-IDPSO, the service composition solution found by the IDPSO algorithm has the smallest fitness value thanks to the mechanism set up to prevent the premature. The fitness value found by PK-IDPSO is the best among all the algorithms. It is because that the PK-IDPSO algorithm benifits from both the service pattern and the IDPSO algorithm. The service pattern greatly reduces the search space of the IDPSO algorithm, resulting in an improved optimization capability.

Convergence

Convergence indicates the speed of algorithm optimization, and the results are shown in Figs. 9, 10, 11, and 12 the influence of iteration times on optimality under different numbers of subtasks.

It can be seen from Figs. 9, 10, 11, and 12 that for different numbers of subtasks, the average optimal values found by various algorithms show different trends as the number of iterations increases. Regardless of the number of subtasks, in addition to the PK-IDPSO algorithm, the IDPSO algorithm has the fastest convergence speed. This is because the premature prevention mechanism we set has a great effect on the convergence. In addition, PK-IDPSO has the fastest convergence rate, and its convergence rate far exceeds that of IDPSO algorithm. This is because the PK-IDPSO algorithm integrates the advantages of the service pattern on top of the advantages of the IDPSO algorithm. The introduction of the service pattern greatly reduces the search space of the IDPSO algorithm, resulting in a greatly improved convergence speed. Therefore, the combination of these two aspects leads to a very high convergence rate of the PK-IDPSO algorithm, which is the highest among all the algorithms discussed in this paper.

Time complexity

Time complexity indicates the performance of the algorithm in time. We set the candidate set corresponding to each subtask to 100 units. Tests are performed with different numbers of subtasks. The results are compared in Table 6 and Fig. 13, where the execution time unit is in seconds (s).

From Table 6 and Fig. 13, it can be concluded that the time performance of the IDPSO algorithm is not very good. This is due to the addition of the escape from the local optimal mechanism, which leads to an increase in its calculation time. However, PK-IDPSO embodies a great time performance advantage. This is because the PK-IDPSO algorithm introduces a service pattern on the basis of the IDPSO algorithm. The introduction of the service pattern simplifies the scale of the problem, so its calculation time is greatly reduced. . When the number of subtasks is small, for example, the number of subtasks is 4. Although the execution time of the PK-IDPSO algorithm is the smallest, it is not much different from the execution time of most comparison algorithms. When the number of subtasks gradually increases, the execution time of IDPSO and the execution time of all comparison algorithms are getting bigger and bigger. The execution time of PK-IDPSO algorithm is much smaller. When the PK-IDPSO algorithm is executed, the particle swarm algorithm IDPSO is not called first, but the service pattern matching algorithm SP- Mapping to match the service patterns, calling the service pattern matching algorithm SP-Mapping will consume a part of the time, and then call the particle swarm algorithm IDPSO to perform instant service combination on the subtasks that are unsuccessful in the service pattern matching. When the number of subtasks is small, call The service pattern matching algorithm SP-Mapping consumes a relatively large proportion of the total time, and when the number of subtasks is small, the execution time of all swarm intelligence optimization algorithms is very small, so the execution time of each algorithm is very close. When the number of sub-

Table 5 Comparison of optimality values of different algorithms with different task numbers

TasksNum	IDPSO	PK-IDPSO	ZP-PSO	S-ABC	EO	RSA
2	0.018623	**0.015802**	0.045560	0.033410	0.028431	0.046875
4	0.023804	**0.018921**	0.048970	0.038906	0.033876	0.051211
6	0.036513	**0.022951**	0.071823	0.056318	0.043113	0.071944
8	0.044916	**0.028045**	0.072481	0.057911	0.047873	0.073264
10	0.048867	**0.030260**	0.073652	0.059221	0.051238	0.072656
12	0.051596	**0.032034**	0.076673	0.060945	0.053893	0.080675
14	0.053529	**0.03311**3	0.078875	0.063467	0.057847	0.083764
16	0.058463	**0.036682**	0.081768	0.065797	0.059321	0.085241
18	0.065115	**0.042794**	0.086387	0.077837	0.066234	0.089243
20	0.069979	**0.043767**	0.089124	0.080346	0.072826	0.091223

	GA	FPA	HAA-ACO	HHO	BBO	HMM-ACO
2	0.032141	0.047215	0.025332	0.035403	0.038192	0.031709
4	0.035908	0.053143	0.028477	0.039129	0.043817	0.041890
6	0.045832	0.061901	0.039453	0.045865	0.057098	0.047075
8	0.051897	0.074091	0.046785	0.059802	0.061023	0.055907
10	0.053132	0.075983	0.050274	0.061971	0.065911	0.061703
12	0.057456	0.081280	0.052011	0.062998	0.071905	0.064763
14	0.059241	0.083987	0.054323	0.064890	0.075907	0.067170
16	0.061836	0.087012	0.059112	0.067131	0.078809	0.069971
18	0.065664	0.089193	0.065901	0.075924	0.081287	0.071378
20	0.075980	0.092124	0.073017	0.081932	0.085230	0.074915

Bold values indicate the smallest fitness value of different algorithms with different task numbers

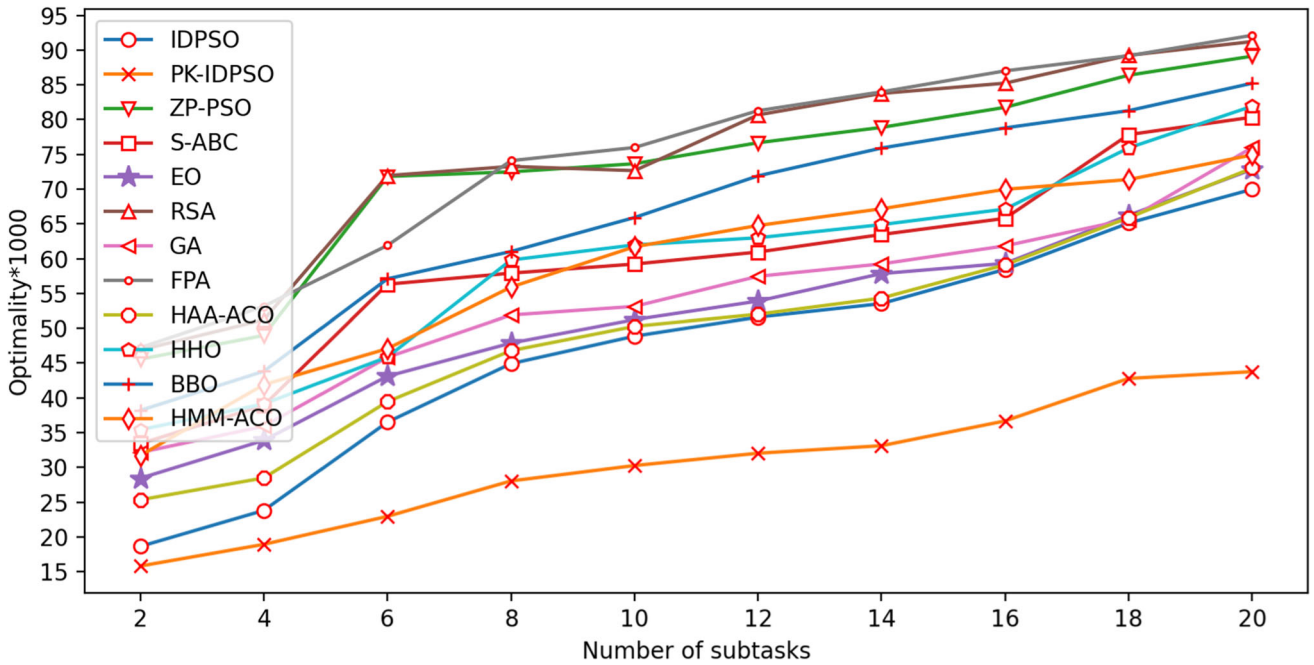

Fig. 8 Comparison of optimality values of different algorithms under different task numbers

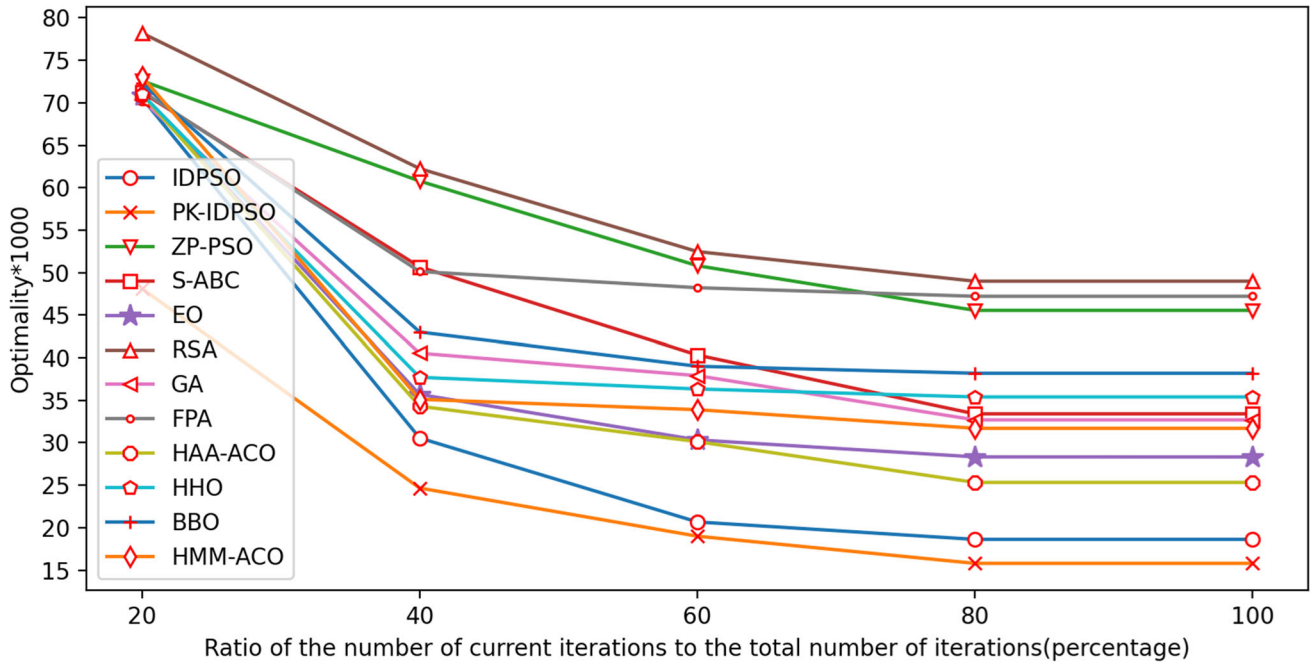

Fig. 9 The influence of the number of iterations on optimality under 2 subtasks

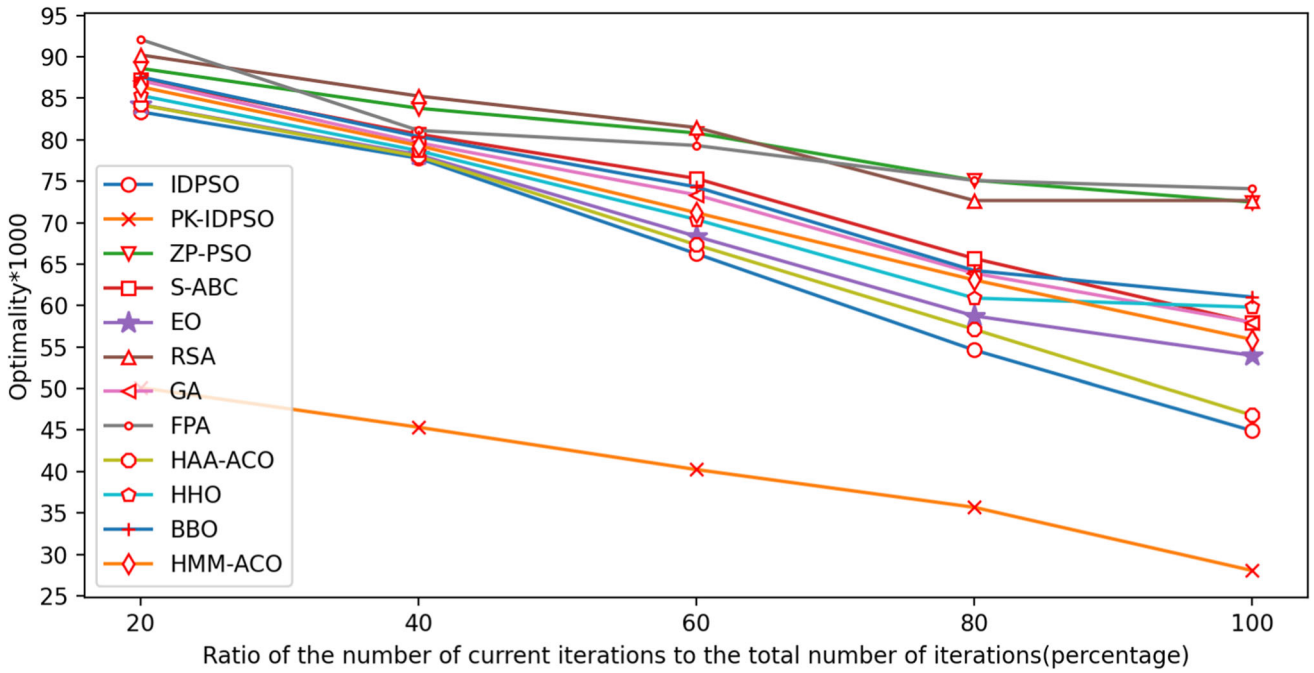

Fig. 10 The influence of the number of iterations on optimality under 8 subtasks

tasks gradually increases, the execution time of each group intelligent optimization algorithm becomes larger and larger. At this time, the time consumed by the PK-IDPSO algorithm to call the SP-Mapping algorithm accounts for a small proportion of the total time, or even negligible. After the service pattern matching is performed, the number of remaining subtasks that call the IDPSO algorithm becomes much less, so the execution time of the IDPSO algorithm will be reduced a lot, and the execution time of the PK-IDPSO algorithm will become much smaller, as shown in Fig. 14, this advantage can be clearly shown. In Fig. 14, the time complexity of the PK-IDPSO algorithm decreases as the coverage of the service patterns increases, and the construction of the service pattern can greatly reduce the time complexity of the algorithm.

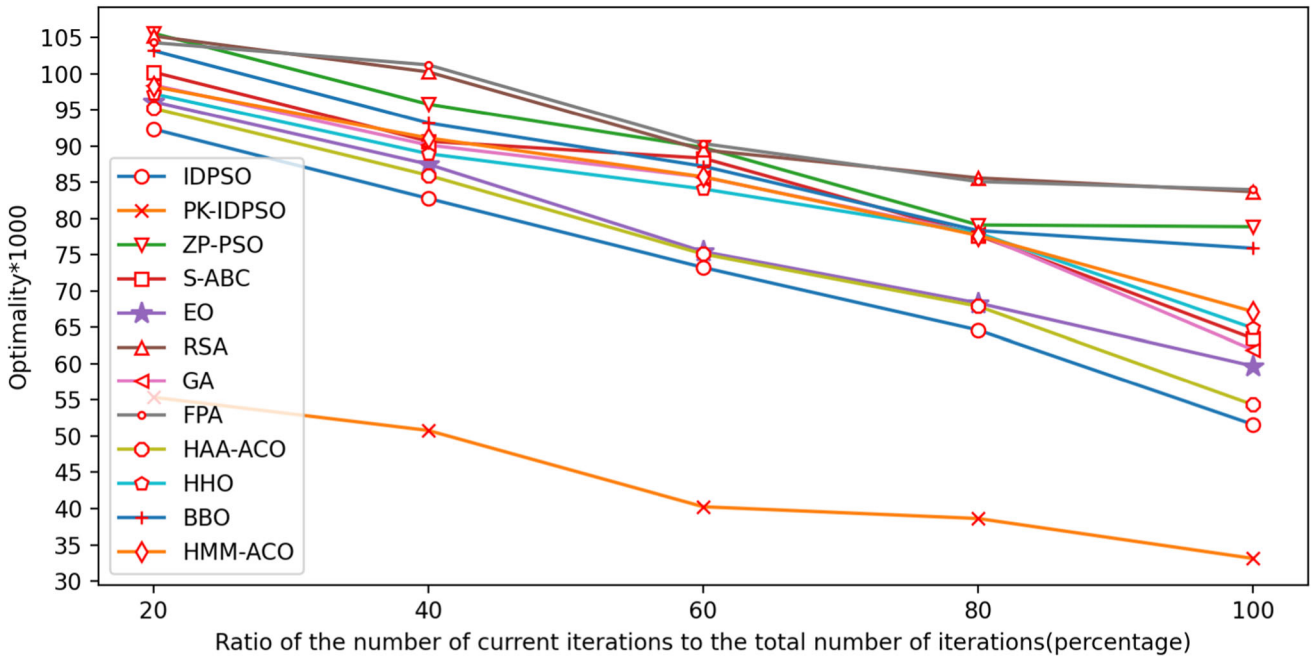

Fig. 11 The influence of the number of iterations on optimality under 14 subtasks

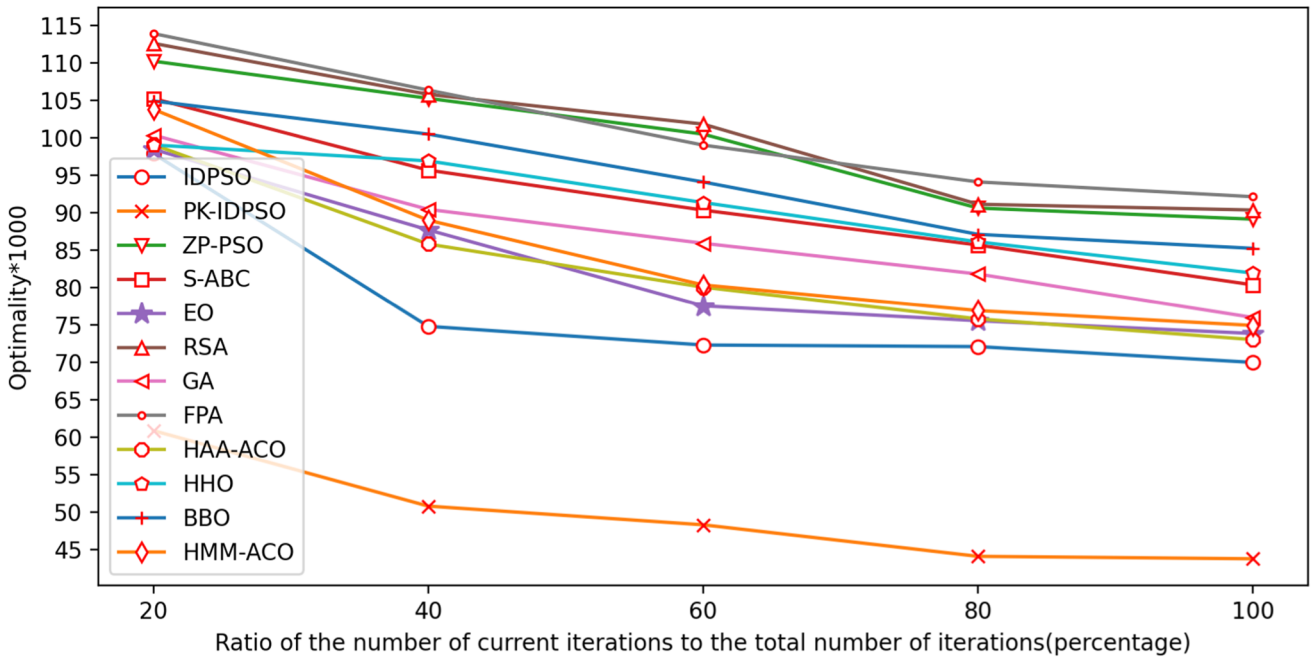

Fig. 12 The influence of the number of iterations on optimality under 20 subtasks

It can be concluded that the PK-IDPSO algorithm can improve the optimization performance regarding the optimality, convergence and time complexity. However, the IDPSO adds the mechanism of avoiding to trap in local optima considered from many aspects, the time complexity is also increased while the optimization performance is improved. In addition, the IDPSO is specially improved for the QoS-SC problem, and may not have such a good optimization ability for other problems. In addition, the PKIDPSO algorithm relies on successfully matched service patterns to have better performance, otherwise its performance may not be guaranteed.

Table 6 Comparison of time complexity of different algorithms under different numbers of subtasks

TasksNum	IDPSO	PK-IDPSO	ZP-PSO	S-ABC	EO	RSA
2	0.0309	**0.0014**	0.0356	0.0302	0.0324	0.0301
4	0.1877	**0.1062**	0.1865	0.1806	0.1803	0.1673
6	1.0141	**0.6309**	1.0031	0.9987	1.0873	0.9152
8	8.7725	**4.7119**	8.6554	8.2336	8.5641	8.0231
10	35.4751	**21.2986**	33.5643	30.6574	34.8752	29.63124
12	85.2681	**51.3869**	80.2145	79.3367	83.6251	77.6733
14	175.3201	**90.0506**	160.3576	155.9017	171.3834	150.3217
16	246.0475	**134.8976**	233.9001	220.9077	247.9567	213.7613
18	567.1813	**289.3534**	540.2231	508.7039	553.9541	473.6253
20	1408.9678	**783.9279**	1189.7903	1120.9087	1383.6095	1047.5762
	GA	FPA	HAA-ACO	HHO	BBO	HMM-ACO
2	0.0357	0.0312	0.0359	0.0300	0.0331	0.0322
4	0.1723	0.1801	0.1875	0.1713	0.1807	0.1890
6	1.0143	0.9857	1.0895	0.9346	1.0797	1.0347
8	8.6324	8.1987	8.6172	8.1902	8.6135	8.9734
10	34.5734	30.1287	35.0135	29.8012	34.0281	36.7783
12	83.7531	78.8907	82.9704	77.9017	81.2103	88.6729
14	165.3231	153.9910	168.7026	152.0838	162.5062	180.4529
16	241.9763	218.9081	243.7750	217.5210	230.1192	251.5436
18	550.2149	493.2375	548.9015	481.7903	543.2019	570.8039
20	1127.8721	1117.8019	1301.2047	1103.9132	1192.3671	1479.8019

Bold values indicate the best time performance value of different algorithms with different task numbers.

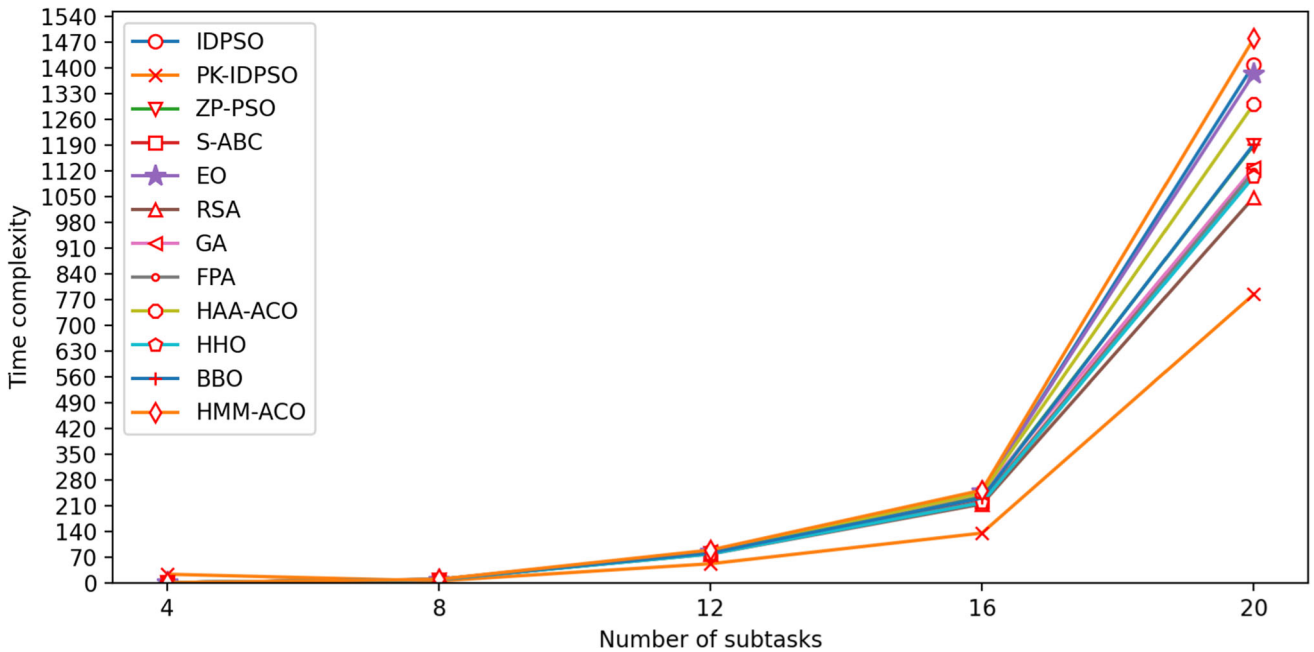

Fig. 13 Comparison of time complexity of different algorithms under different numbers of subtasks

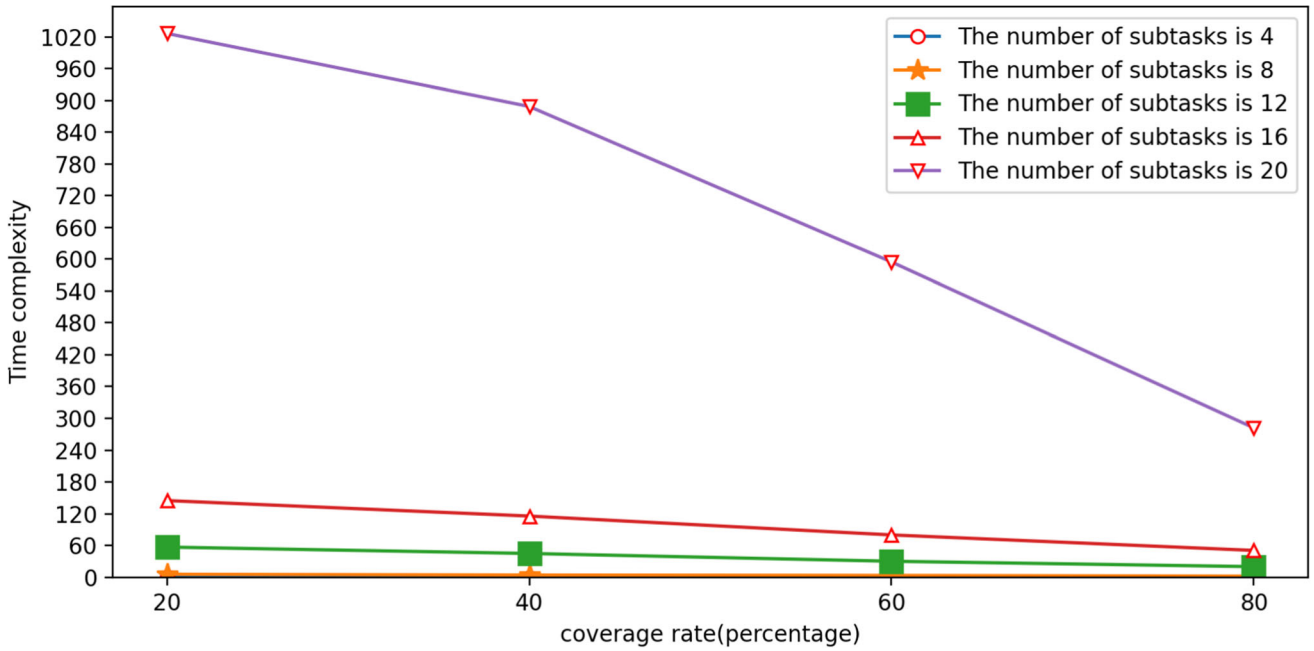

Fig. 14 The relationship between PK-IDPSO time complexity and service pattern coverage under different numbers of subtasks

Conclusion

In this work, the construction of service patterns based on prior knowledge is studied and the application of improved discrete particle swarm optimization in QoS-SC optimization problems is investigated. The SP-Mapping, IDPSO, and PK-IDPSO algorithms are proposed, which effectively solve the traditional swarm intelligence problems of trapping in local optima in the field of service composition optimization.This paper firstly constructs service patterns based on domain knowledge. Then the particle swarm algorithm is improved, we propose four methods to prevent particle prematurity, including sorting the candidate service set, resetting the particle position after a certain algebra, setting the control parameters of the particle swarm algorithm reasonably, and setting the optimality of reference, which enhances the global optimization ability of the particle swarm optimization algorithm, and effectively solves the problem that a particle population easily traps in local optima. When a service composition request is received, the method in this paper firstly calls the SP-Mapping algorithm to match service patterns to obtain a partial solution, and then calls the IDPSO algorithm for some requests that are not matched successfully, and finally obtains a complete solution. The experimental results show that the prior knowledge-based service composition algorithm PK-IDPSO proposed outperforms the state-of-the-art methods tested.

Compared with other swarm intelligence service composition algorithms, the method proposed in this paper has two strenghts: (1) the use of domain prior knowledge to guide service composition attempts to reduce the search space of swarm intelligence algorithms, so it can greatly improve the search efficiency of swarm intelligence algorithms, and service patterns mining using domain prior knowledge has been proved to be excellent by a large number of users, so it can well meet the needs of users. (2) the IDPSO algorithm proposed in this paper is improved according to the characteristics of the QoS-SC problem, so it has better performance for the QoS-SC problem.

However, the method proposed in this paper has the following limitations: First, the control parameters of the IDPSO algorithm are not dynamically set. Second, the method of service pattern mining is relatively simple, and the mining of service patterns requires historical service records in a specific format, which leads to its limited practical application temporarily. In view of the above limitations, further efforts would be made in future work: (1) use the deep learning algorithm to realize the automatic setting of the control parameters of the IDPSO algorithm. (2) the natural language technology could be used to process complex service history records of different formats, and the knowledge graph technology could be used to mine more complex and obscure service patterns on this basis.

Acknowledgements This work is supported by the National Key R&D Program "Research and Development of Collaborative Technology and Platform" under Grant No. 2018YFB1402900.

References

Abualigah, L., Diabat, A., Sumari, P., & Gandomi, A. H. (2021). Senior member. Applications, deployments, and integration of internet of drones (IoD): A review. *IEEE Sensors Journal*, *21*(22).

Abualigah, L., Elaziz, M. A., Sumari, P., Geem, Z. W., & Gandomi, A. H. (2022). Reptile search algorithm (RSA): A nature-inspired meta-heuristic optimizer. *Expert Systems with Applications, 191,* 116158.

Abualigah, L., Yousri, D., Elaziz, M. A., Ewees, A. A., Al-qaness, M. A. A., & Gandomi, A. H. (2021). Aquila optimizer: A novel meta-heuristic optimization algorithm. *Computers & Industrial Engineering, S036–08352*(21), 00154–00156.

Cergibozan, A. C., & Tasan, S. (2022). Genetic algorithm based approaches to solve the order batching problem and a case study in a distribution center. *Journal of Intelligent Manufacturing, 33,* 137–149.

Chen, F., Dou, R., Li, M., & Harris, W. (2016). A flexible QoS-aware web service composition method by multi-objective optimization in cloud manufacturing. *Computers & Industrial Engineering, 99,* 423–431.

Chen, Y., Huang, J., & Lin, C. (2014). Particle selection: An efficient approach for QoS-aware web service composition. In *IEE international conference on web services* (pp. 1–8).

Cook, W., Held, S., & Helsgaun, K. (2021). Constrained local search for last-mile routing. *Cornell University*. https://doi.org/10.48550/arXiv.2112.15192.

Eberhart, R., & Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on micro machine and human science* (pp. 39-43).

Falch, M., Idongesit, W., & Reza, T. (2020). Cross-border provision of e-Government business registration services. In *International telecommunications society (ITS)*. ITS Conference, Online Event 224852.

Gao, Z., Zhao, J., Yurong, H., & Chen, H. (2021). The challenge for the nature-inspired global optimization algorithms: Non-symmetric benchmark functions. *IEEE Acess, Digital Object Identifier*. https://doi.org/10.1109/ACCESS.2021.3100365.

Guo, X., Chen, S., Zhang, Y., & Li, W. (2018). Application of fireworks particle Swarm optimization algorithm in web service composition. *Journal of Chinese Computer Systems, 39*(6), 1312–1316.

Hongwei, L., Wei, K., & Kong, H. (2006). An improved high-effictive KMP pattern matching algorithm. *Huazhong University of Science & Technology (Nature Science Edition), 34*(10), 41–43.

Huo, Y., Qiu, P., Zhai, J., Fan, D., & Peng, H. (2017). Multi-objective service composition model based on cost-effective optimization. *Applied Intelligence, 48*(3), 651–669.

Huo, Y., Zhuang, Y., Jingjing, G., Ni, S., & Yu, X. (2015). Discrete gbest-guided artificial bee colony algorithm for cloud service composition. *Applied Intelligence, 42*(4), 661–678.

Jatoth, G. R. C., & Gangadharan, U. F. (2019). Optimal fitness aware cloud service composition using modified invasive weed optimization. *Swarm and Evolutionary Computation, 44,* 1073–1091.

Jin, H., Lv, S., Yanga, Z., & Liu, Y. (2022). Eagle strategy using uniform mutation and modified whale optimization algorithm for QoS-aware cloud service composition. *Applied Soft Computing, 114,* 108053.

Jin, H., Yao, X., & Chen, Y. (2015). Correlation-aware QoS modeling and manufacturing cloud service composition. *Journal of Intelligent Manufacturing*, 114.

Kashyap, A., Kumari, C., & Chhikara, R. (2020). Service composition in IoT using genetic algorithm and particle swarm optimization. *Open Computer Science, 10,* 56–64.

Khanouche, M. E., Attal, F., Amirat, Y., Chibani, A., & Kerkar, M. (2019). Clustering-based and QoS-aware services composition algorithm for ambient intelligence. *Journal of Information Science, 482,* 419–439.

Klai, K., & Ochi, H. (2016). Model checking of composite cloud services. In *IEEE international conference on web services* (pp. 356–363)

Kurokawa, S., & Matsui, T. (2021). Dynamic programming and linear programming for odds problem. *Cornell University*.

Li, C. Y., Li, J., Chen, H. L., & Heidari, A. A. (2021). Memetic Harris Hawks optimization: Developments and perspectives on project scheduling and QoS-aware web service composition. *Expert Systems with Applications, 171,* 114529.

Li, T., He, T., Liu, Y., Wang, Z., & Zhang, Y. (2020). SDF-GA: A service domain feature-oriented approach for manufacturing cloud service composition. *Journal of Intelligent Manufacturing, 31,* 681–702.

Liang, H., Wen, X., Liu, Y., Zhang, H., Zhang, L., & Wang, L. (2021). Logistics-involved QoS-aware service composition in cloud manufacturing with deep reinforcement learning. *Robotics and Computer Integrated Manufacturing, 67,* 101991.

Liu, C., Wan, Z., Liu, Y., Li, X., & Liu, D. (2021). Trust-region based adaptive radial basis function algorithm for global optimization of expensive constrained black-box problems. *Applied Soft Computing, 105,* 107233.

Liu, R., Wang, Z., & Xiaofei, X. (2019). Parameter tuning for S-ABCPK an improved service composition algorithm considering priori knowledge. *International Journal of Web Services Research, 16*(2), 88–109.

Liu, R., Xu, X., Wang, Z., Sheng, Q.Z., & Xu, H. (2017). Probability matrix of request-solution mapping for efficient service selection. In *2017 IEEE 24th international conference on web services*. IEEE. https://doi.org/10.1109/ICWS.2017.51.

Maatouk, O., Ayadi, W., Bouziri, H., & Duval, B. (2021). Evolutionary algorithm for the biclustering of gene expression data based on biological knowledge. *Applied Soft Computing Journal, 104,* 107177.

Mabrouk, N. B., Beauche, S., Kuznetsova, E., Georgantas, N., & Issarny, V (2009). QoS-aware service composition in dynamic service oriented enviroments. In *Middleware 2009* (pp. 123–142). Springer.

Momeni, K. (2021). Service integration: Supply chain integration in servitization. *Springer*. https://doi.org/10.1007/978-3-030-75771-7_30

Sailer, J. (2014). M2M internet of things web of things industry 4.0. *Elektrotechnik & Informationstechnik, 131*(1), 3–4.

Sangaiah, A. K., Bian, G.-B., Bozorgi, S. M., Suraki, M. Y., Hosseinabadi, A. A. R., & Shareh, M. B. (2020). A novel quality-of-service-aware web services composition using biogeography-based optimization algorithm. *Soft Computing, 24,* 8125–8137.

Sefati, S., & Navimipour, N. J. (2021). A QoS-aware service composition mechanism in the internet of things using a hidden-Markov-model-based optimization algorithm. *IEEE Internet of Things Journal, 8*(20), 15620–15627.

Seghir, F., & Khababa, A. (2018). A hybrid approach using genetic and fruit fly optimization algorithms for QoS-aware cloud service composition. *Journal of Intelligent Manufacturing, 29*(8), 1773–1792.

Vouk, M. A. (2008). Cloud computing issues, research and implementations. *Journal of Computing and Information Technology-CIT16,4*, 235–246.

Wang, Y., Wang, S., Yang, B., Gao, B., & Wang, S. (2022). An effective adaptive adjustment method for service composition exception handling in cloud manufacturing. *Journal of Intelligent Manufacturing, 33,* 735–751.

Wen, T., Sheng, G., Guo, Q., & Li, Y. (2013). Web service composition based on modified particle swarm optimization. *Chinese Journal of Computers, 36*(5), 1031–1046.

Xu, H., Li, N., Wang, X., Xu, X., Wang, Y., Tu, Z., & Wang, Z. (2020). Domain priori knowledge based integrated solution design for

internet of services. In *2020 IEEE international conference on services computing (SCC)*. IEEE.

Zeng, L., Benatallah, B., & Dumas, M. (2003). Quality driven web services composition. In *Proceedings of the 12th international conference on world wide web* (pp. 411–421).

Zeng, L., Benatallah, B., Ngu, A. H. H., Dumas, M., Kalagnanam, J., & Chang, H. (2004). QoS-aware middleware for web services composition. *IEEE Transactions on Software Engineering, 30*(5), 311–327.

Zhang, B., Wen, K., Jianhua, L., & Zhong, M. (2021). A top - k QoS-optimal service composition approach based on service dependency graph. *Journal of Organizational and End User Computing, 33*(3), 50–68.

Zhang, S., Yangbing, X., Zhang, S., Zhang, W., & Dejian, Yu. (2019). A new fuzzy QoS-aware manufacture service composition method using extended flower pollination algorithm. *Journal of Intelligent Manufacturing, 30*(5), 2069–2083.

Zhang, Y., Cui, G., Deng, S., Chen, F., Wang, Y., & He, Q. (2021). Efficient query of quality correlation for service composition. *IEEE Transactions on Services Computing, 14*(3), 695–709.

Zhang, Y., Gui, G., Yan, Y., Zhao, S., & Zhao, Y. (2018). Quality constraints-aware service composition based on task granulating. *Journal of Computer Research and Devolopment, 55*(6), 1345–1355.

Zhang, Y., Jing, Z., & Zhang, Y. (2015). MR-IDPSO: A novel algorithm for large-scale dynamic service composition. *Singhua Science and Technology, 20*(6), 62–612.

Zhang, Z. (2020). Big data service in distributed network environment based on FPGA. *Microprocessors and Microsystems*. https://doi.org/10.1016/j.micpro.2020.103586.

Zhou, J., & Yao, X. (2017). A hybrid approach combining modified artificial bee colony and cuckoo search algorithms for multi-objective cloud manufacturing service composition. *International Journal of Production Research, 55*(16), 4765–4784.