SDF-GA

Li, Tianyang; Ting, He; Wang, Zhongjie; Zhang, Yufeng

SDF-GA: a service domain feature-oriented approach for manufacturing cloud service composition

Tianyang Li [1] · Ting He[1,2] · Zhongjie Wang[1] · Yufeng Zhang[3]

[1]School of Computer Science and Technology, Harbin Institute of Technology, Harbin, Heilongjiang 150001, China

[2]College of Computer Science and Technology, Huaqiao University, Xiamen, Fujian 361021, China

[3]Birmingham Business School, University of Birmingham, Birmingham B15 2TT, UK

E-mail: xuantinghe@gmail.com; tianyangli1985@gmail.com

Corresponding author: Ting He

Abstract (150-250 words): Cloud manufacturing (CMfg) is a new service-oriented manufacturing paradigm in which shared resources are integrated and encapsulated as manufacturing services. When a single service is not able to meet some manufacturing requirement, a composition of multiple services is then required via CMfg. Service Composition and Optimal Selection (SCOS) is a key technique for efficient manufacturing service composition to create an on-demand Quality of Service (QoS) to satisfy various user requirements. Given the number of services with the same functionality and a similar level of QoS, SCOS has been seen as a key challenging research area in CMfg. One effective approach for solving SCOS problems is to use Service Domain Features (SDF) through investigating the probability of services being used for a specific requirement from multiple angles. The approach can result in a division of service space, and then help narrow down the service space with large-scale candidate services. The approach can also search for optimal subspaces that most likely contribute to an overall optimal solution. In doing so, this paper develops an SDF-oriented genetic algorithm to effectively create a manufacturing service composition with large-scale candidate services. Fine-grained SDF definitions are developed to divide the service space. SDF-based optimization strategies are adopted. The novelty of the proposed algorithm is presented based on Bayesian theorem. The effectiveness of the proposed algorithm is validated through solving three real-world SCOS problems in a private CMfg.

Key words: Service domain features, Service composition and optimal selection, Cloud manufacturing, manufacturing cloud service composition, genetic algorithm

1 Introduction

With the application of cloud computing, Internet of Things, service computing, virtualization and other information and communication technologies, cloud manufacturing (CMfg) has gradually realized enterprise applications for creating value-added services and increasing competitiveness (Pisching et al. 2015; Tao et al. 2017; Ren et al. 2017). As a new type of service-oriented manufacturing paradigm, all types of manufacturing resources are encapsulated in CMfg as manufacturing services for enabling complete sharing and integration (Li et al. 2010; Zhang et al 2017b, Wang et al 2017). To satisfy various user requirements on demand and realize the efficient use of enterprise resources, one of the key technologies in implementing CMfg is Service Composition and Optimal Selection (SCOS), which involves aggregating various services with different functionalities into a value-added compositional service to address complex manufacturing tasks (Tao et al. 2013; Kubler et al. 2016). According to the given user requirements and constraint conditions, the first step of SCOS process in CMfg is to decompose the request into a composed service execution path with subtasks. Then, candidate service sets for each subtask are collected. Finally, a corresponding service for each subtask is selected to composite an optimal solution for satisfying user requests. Undoubtedly, building such a Quality of Service (QoS)-optimal solution is a typical multi-criterion nondeterministic polynomial (NP)-hard problem (Tao et al. 2013; Zhou and Yao 2017).

Because the manufacturing request is personalized and complicated, the execution path of SCOS becomes long and complex. In addition, because there is a sharp increase in manufacturing resources in the service pool, an increasing number of manufacturing services are required to participate in the long and complex execution path (Fatahi and Houshmand 2014; Xiang et al. 2016; Xue et al. 2016). Indeed, the increasing complexity of manufacturing requests and vast amounts of service candidates will significantly affect the stability of a CMfg system. In an extreme situation, without an effective algorithm for solving SCOS

problems, CMfg can collapse when concurrent, massive numbers of users request plentiful service compositions from a mass of alternative candidate services with similar functionality and QoS in CMfg (Tao et al. 2015). Moreover, Service Domain Features (SDFs) (including prior, correlation, and similarity) (Xu et al. 2017), which are important objective regulators in the service domain, have gradually shaped service applications and evolution in services and manufacturing industries over the long term. These SDFs reflect the substantive characteristic and applied probability of services used to meet a certain type of requirement and strongly accelerate the solving of SCOS problems (Xu et al. 2017). Unfortunately, SDFs have been used insufficiently and unreasonably to design algorithms for solving SCOS problems (Zhou and Yao 2017), and the solving of SCOS problems is still a key challenge (Zhang et al 2017a).

To overcome these gaps, an SDF-oriented method must first be created to effectively and efficiently solve SCOS problems (Liu and Xu 2014). This method should take advantage of the fully mined value of SDFs to properly narrow the service space with large-scale candidate services and to richly search optimal subspaces that most likely contain the global optimal solution. Despite the importance of developing SDF-oriented methods for SCOS, the literature only refers to single SDF or fragmentary SDFs to roughly divide the service space and achieve inefficient optimization strategies for obtaining the global optimal solution, which further causes inefficiencies in solving SCOS problems and inconsistencies between theoretical studies and practical applications.

No comprehensive method exists in the literature that richly investigates SDFs and applies them to the design of effective algorithms for solving SCOS. The lack of literature precedent raises three questions: 1) what definitions of SDFs are used to process the large-scale service space of SCOS in CMfg? 2) How can SDFs be reasonably used to improve the local and global search capabilities of the algorithms? 3) What are the properties of SDF-oriented algorithms and their performance boundaries for solving SCOS?

This paper thus proposes an innovative SDF-oriented genetic algorithm (SDF-GA) that can be used to implement CMfg for effectively solving SCOS problems. The goal is to provide an algorithm to solve SCOS problems with theoretical insights based on SDFs, particularly for large-scale manufacturing cloud service compositions. The effectiveness and efficiency of the proposed algorithm have been validated with three application examples and theoretical analysis based on the Bayesian theorem. By exploiting SDFs for SCOS problems, this work makes novel contributions for CMfg application and realization and extends the theory of SDF-oriented intelligence optimization.

The paper is structured as follows. Section 2 provides an overview of related SCOS works and illustrates current problems. Section 3 introduces the formulation of SCOS problems and their general mathematical model. Definitions of SDFs and an algorithm for dividing the service space are detailed in Section 4. Section 5 elaborates the SDF-oriented GA with optimization strategies, and the superiority of the SDFs is analysed using the Bayesian theorem. The effectiveness of the SDF-GA is verified with three case studies in Section 6. Finally, Section 7 concludes the paper and provides future research directions.

2 Related work

The SCOS problem originates from the web service and cloud computing areas and aims at reducing the size of candidate services and selecting appropriate ones from the remaining services for a composition, thereby improving the efficiency and quality of service compositions (Tao et al. 2014b; Lemos et al. 2016). In CMfg, various types of manufacturing services with different QoSs are continually aggregating to form a large service pool and large search space that must be addressed to achieve the global optimal solution. Hence, SCOS problems in CMfg are more complicated (Xue et al. 2016; Morgan and O'Donnell 2017).

For this NP-hard problem in CMfg, the most widely used approaches are improved heuristic algorithms such as the Traditional Genetic Algorithm (T-GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Artificial Bee Colony (ABC) with context-aware (Zhou and Yao 2017), QoS-aware (Huang et al. 2014; Zheng et al 2016; Seghir and Khababa 2016; Chen et al. 2016), and semantic-web-based (Lu and Xu 2017) techniques. These works have furthered the research on SCOS problems in CMfg. However, they are inefficient at solving large-scale SCOS problems because they do not consider the SDFs (Zhou and Yao 2017).

Actually, due to the significant influence of SDFs on solving SCOS problems, SDFs have been widely employed to solve SCOS problems. Some typical research efforts that have used SDFs to boost the performance of algorithms for solving SCOS are summarized in Table 1.

Table 1 Summary of SDF-based methods

The SDF	Current methods	Advantages	Disadvantages
Prior	1. Employ recorded execution sequential patterns or execution dependency of service to select services for an SCOS problem (Zhang et al. 2008; Liu et al. 2016) 2. Initialize current optimization algorithms with a historical solution of a request that is similar to the current user requests for solving SCOS problems (Xiang et al. 2016)	1. Reuse historical patterns or solutions to efficiently solve SCOS problems	1. Use a single feature 2. Only improve the local search capability of algorithms 3. Do not consider new services for solving SCOS problems
Correlation	1. Employ the business correlations of services as the key factor to negotiate the choice of concrete services for service composition (Wu et al. 2014; Van et al. 2015) 2. Employ an auxiliary graph to express the QoS correlation-aware service composition, and design a fast algorithm to search optimal solutions (Hua et al. 2014) 3. Develop formalized description for the QoS correlation between two services and design algorithms to discovery correlations to apply them to service composition (Hua et al. 2014; Tao et al. 2010; Jin et al. 2017)	1. Use historical schemes based on business and QoS correlations to facilitate the solving of SCOS problems	1. Use a single feature 2. Only improve the local search capability of algorithms 3. Do not consider new services and high-frequency services in solving SCOS problems
Similarity	1. Define similarity measures to determine the closeness of a historical composition solution with respect to any new request to reuse existing solutions (Bravo 2014) 2. Employ similarity rating of friendship, social contact, and community of interest relationships as the filter to select concrete services for service recommendation (Chen et al. 2016) 3. Employ the internal features of services and end users, such as locations, configurations, functionality, and user profiles, to calculate similarity of services and then predict the end-to-end QoS values of services for composite service (Karim et al. 2015) 4. Establish a framework to use QoS time series inter-correlations and apply a novel time-series group similarity approach to predict QoS values for composite service (Ye et al. 2016)	1. Use historical solutions and their substitutions to facilitate the solving of SCOS problems	1. Use a single feature 2. Only improve the local search capability of algorithms 3. Do not use services with outstanding QoS values for solving SCOS problems
SDFs (including prior, correlation and similarity)	1. Define the main SDFs and analyse the influences of these SDFs in solving SCOS toward proposing a new Service domain-oriented Artificial Bee Colony algorithm paradigm (S-ABC) based on the optimization mechanism of ABC (Xu et al. 2017) 2. Develop a context-aware and Differential Evolution-enhanced Artificial Bee Colony (DE-caABC) algorithm based on three SDFs and divisions of the service space for solving SCOS in CMfg (Zhou and Yao 2017)	1. Narrowed service space 2. Efficient algorithms based on the influence mechanism of SDFs	1. Coarse-grained definitions of SDFs and insufficient division of service space 2. Does not use services with outstanding QoS values 3. Limited improvement of capabilities of algorithms 4. Lack theoretical analysis and insights

As seen from the table, single-feature-oriented algorithms can efficiently solve SCOS problems. However, without combining other features and considering new and outstanding services, the improvement in the search capability of these algorithms is quite limited. Studies on S-ABC and DE-caABC have demonstrated the superiority of SDFs in improving the performance of algorithms for solving SCOS problems. However, with coarse-grained definitions of SDFs, an insufficient division of the service space and the lack of theoretical analysis in these works, the potential of SDFs for SCOS problems remains to be fully developed. In this paper, we investigate SDFs more deeply and make the division of the service space more reasonable. Combined with appropriate optimization strategies and theoretical analysis, we attempt to design a more effective and efficient algorithm to solve SCOS problems with a large-scale service space in CMfg.

3 Problem description and mathematical model

A service request in CMfg is sequential and personalized (Xue et al. 2016). When a request is submitted to a CMfg platform, it is first pre-processed and transferred to a manufacturing task. Then, the task is decomposed into several subtasks in a service process. Next, the platform gathers a candidate service set for each subtask. Finally, only one service can be optimally selected

from each candidate service set based on multiple objectives or a single objective to be invoked in sequence to construct a composite service execution path. As shown in Fig. 1, such a SCOS process contains three steps:

(1) Requirement processing and task decomposition: When a user submits a personalized request to the CMfg service, the request is first clustered into a requirement class c_i and transferred to a manufacturing task t (requests in the same class have a similar task). Simultaneously, a set of QoS attributes $Q(t)$ about the manufacturing task is identified, and the corresponding set of constrained values of QoS attributes $q(t)$ is determined. Thereafter, the task requirement is divided into a set of subtasks st to form a service process SP based on the composite service execution path. Accordingly, there are QoS requirements $q(st_i)$ for each subtask with the decomposing of task requirements. In this paper, we define $c_i=\{r_1, \ldots, r_m\} \in C=\{ c_1, c_2, \ldots, c_d\}$, where r_m represent a certain request, C is the set of requirement classes, and d is the number of classes. $SP=\{st_1, st_2, \ldots, st_j\}$, where j is the number of subtasks; $Q(t)=\{Q_1, \ldots Q_n\}$; $q(t)=\{q_{10}, \ldots, q_{n0}\}$; and $q(st_j)=\{q_{1j}, q_{2j}, \ldots, q_{nj}\}$, where n is the number of attributes, and q_{n0} and q_{nj} are the constraint values of the nth QoS attributes for t and st_j, respectively.

(2) Matching candidate services for each subtask: For each subtask st_j, CMfg will utilize matching algorithms to retrieve multiple services that satisfy the functional and QoS requirements $q(st_j)$. The qualified manufacturing services (MSs) are combined into a set of candidate services for each subtask. For a specific subtask st_j, its corresponding candidate service set is $MS(st_j) = \{MS_{j1}, MS_{j2}, \ldots, MS_{jk}\}$, where MS_{jk} is the jth MS for st_j.

(3) Service composition and optimal selection: For each subtask, hundreds of services that have the same functionality and different values of QoS properties can exist. Assuming that a task contains j subtasks and that each subtask has h candidate MSs, there will be h^j feasible solutions. This problem is NP-complete and needs to be solved with the global optimal solution to meet a user's requirements.

In this paper, we focus on designing an effective algorithm for step 3.

The value of the QoS attribute Q_n of a manufacturing service composition (MSC) is aggregated by the corresponding attribute values of all subtasks. Let $q(MSC1)=\{q_{1,msc}, q_{2,msc}, \ldots, q_{n,msc}\}$ and $q(MS_i)=\{q_{1,i}, q_{2,i}, \ldots, q_{n,i}\}$ express the aggregated QoS values of MSC and the QoS values of a single service belonging to the MSC; then, $q_{n,msc}=f(q_{n,1}, \ldots, q_{n,j})$. f is an aggregation function based on the structure of the composite service execution path. Currently, there are four main workflow patterns in the composite service execution path: sequential, parallel, selective, and circular. The QoS attributes and their aggregation functions for types of workflow patterns used in this paper are shown in Tables 2 and 3, respectively.

Table 2 QoS attributes for SCOS

QoS attributes	Description
Cost	Utility fee of service for each request
Execution time	Time to perform the service functionality
Respond time	The waiting time before execution
Availability	1-failure rate
Reliability	Uptime/(uptime-downtime)

Table 3 QoS aggregation functions for various workflow patterns

QoS attribute	Sequence	Parallel	Switch	Loop
Cost (C)	$\sum_{i=1}^{k} C(MS_i)$	$\sum_{i=1}^{k} C(MS_i)$	$\sum_{i=1}^{k} p_i * C(MS_i)$	$k * C(MS_i)$
Execution time (ET)	$\sum_{i=1}^{k} ET(MS_i)$	$max(ET(MS_i))$	$\sum_{i=1}^{k} p_i * ET(MS_i)$	$k * ET(MS_i)$
Respond time (RT)	$\sum_{i=1}^{k} RT(MS_i)$	$max(RT(MS_i))$	$\sum_{i=1}^{k} p_i * RT(MS_i)$	$k * RT(MS_i)$
Availability (A)	$\prod_{i=1}^{k} A(MS_i)$	$min(A(MS_i))$	$\sum_{i=1}^{k} p_i * A(MS_i)$	$\prod A(MS_i)$
Reliability (R)	$\prod_{i=1}^{k} R(MS_i)$	$min(R(MS_i))$	$\sum_{i=1}^{k} p_i * R(MS_i)$	$\prod R(MS_i)$

There are two types of QoS attributes: positive and negative (Zhou and Yao 2017). Positive attributes mean that the higher

the value of the QoS criteria, the better the quality of the MS, for example, availability and reliability. However, negative attributes mean that the lower the value of the QoS, the better the quality of the MS, for example, execution time and cost. For a single-objective optimization problem of SCOS based on a certain QoS attribute, the target and constraint functions can be defined as follows:

$$\begin{gathered} min/max\, q_{n,msc} = f\left(q_{n,1}, \dots, q_{n,j}\right) \\ subject q_{i,msc \wedge i \neq n} = \varphi\left(q_{i,1}, \dots, q_{i,j}\right) \leq q_{i,0} \\ q_{k,msc \wedge i \neq n} = \varphi\left(q_{k,1}, \dots, q_{k,j}\right) \geq q_{k,0} \end{gathered} \quad (1)$$

where $q_{n,msc}$ is the target QoS attribute of an MSC that must be optimized, and $f\left(q_{n,1}, \dots, q_{n,j}\right)$ is its aggregation function based on values of the corresponding QoS attribute of the selected MSs. If $q_{n,msc}$ is positive, then the maximization function should be used; otherwise, the minimum function should be used. Similarly, $\varphi\left(q_{i,1}, \dots, q_{i,j}\right)$ is the aggregation function for a constrained QoS attribute, and $q_{i,0}$ and $q_{k,0}$ are constrained values.

For a multi-objective optimization problem, the values of positive and negative QoS attributes should be normalized via the following formulas.

$$q_{k,nor-} = \begin{cases} \frac{q_{k,max} - q_{k,msc}}{q_{k,max} - q_{k,min}}, if\, q_{k,max} \neq q_{k,min} \\ 1 \qquad\qquad\quad , if\, q_{k,max} = q_{k,min} \end{cases} \quad (2)$$

$$q_{k,nor+} = \begin{cases} \frac{q_{k,msc} - q_{k,min}}{q_{k,max} - q_{k,min}}, if\, q_{k,max} \neq q_{k,min} \\ 1 \qquad\qquad\quad , if\, q_{k,max} = q_{k,min} \end{cases} \quad (3)$$

where $q_{k,max}$ and $q_{k,min}$ are the minimum and maximum aggregated values of the *k*th QoS criteria of the MSC; $q_{k,max} = \sum_{t=1}^{j} max q_{k,t}$ and $q_{k,min} = \sum_{t=1}^{j} min q_{k,t}$, where max $q_{k,t}$ and min $q_{k,t}$ are the minimum and maximum values of the *k*th QoS criterion from the candidate service set *t*.

Based on the normalized values, a utility function can be used to transform the multi-objective optimization problem into a single maximization optimization problem as follows:

$$\begin{gathered} max f(X) = \sum_{i=1}^{n} w_i q_{i,X} \\ subject q_{i,msc} \geq q_{i,0} \vee q_{i,msc} \leq q_{i,0} \\ q_{k,msc} \leq q_{k,0} \end{gathered} \quad (4)$$

where *f(X)* is the utility function, w_i is the weight for each aggregated value of *QoS* criteria, $q_{i,msc}$ is the aggregated values of the *i*th QoS criterion of the MSC, and $q_{i,0}$ are constrained values for negative and positive QoS attributes.

Commonly, the selection of a multi-objective or single-objective optimization problem for SCOS relies on the given request. In this paper, we embody multi-objective optimization models to verify our proposed algorithm in section 5.

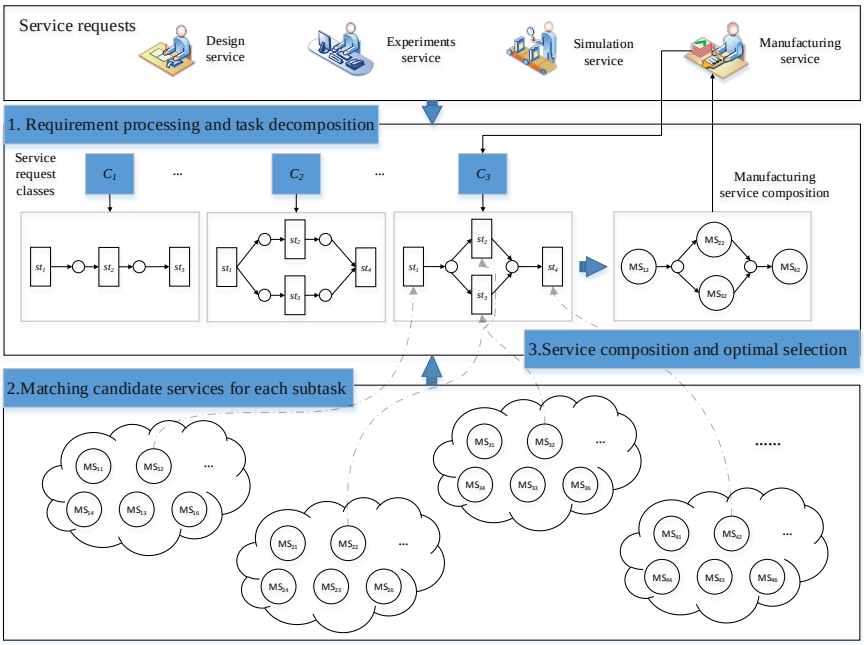


Fig. 1 SCOS process in CMfg

4 Service space preprocessing

4.1 Service domain features

Three features exist in a typical service domain, prior, correlation and similarity, which are defined in the following.

Definition 1 (Prior). The prior is a posterior probability of a candidate service MS_{ij} being used to satisfy a request in c_d. This probability is experiential knowledge abstracted from the service-composition execution record of c_d. From an intuitional perspective, for a specific service request in class c_d, there always exist services or solutions with higher use frequency and user satisfaction. This objective law can be expressed as a potential use probability of $MS_{i,j}$ which can be calculated by the Bayesian formula as follows:

$$P(c_{d+}/MS_{i,j}) = \frac{P(c_{d+})P(MS_{i,j}/c_{d+})}{P(MS_{i,j})} \quad (5)$$

where $P(c_{d+}) = \frac{H^+}{H}$ is the percentage of positive records (records with higher user satisfaction or another criterion) from using the partition function $U(h) \geq T$, $U(h)$ is the partition function for calculating the degree of user satisfaction or another criterion, T is the threshold of $U(h)$, H is the number of complete historical records, H^+ is the number of positive records that satisfy $U(h) \geq T$, H^- =H-H^+ is the number of negative records, $P(MS_{i,j}/c_{d+}) = \frac{N^+}{H^+}$ is the used probability that $MS_{i,j}$ is used in positive records, N^+ is the number of occurrences of $MS_{i,j}$ in positive records, $P(MS_{i,j}) = \frac{N}{H}$ is the probability of $MS_{i,j}$ being used by requests in c_d, and N is the occurrence number of $MS_{i,j}$ in the overall records of c_d.

The prior value of an existing solution in the historical records of c_d can be calculated by the naive Bayes formula. The reuse probability of solution s can be calculated as follows:

$$P(s/v_j) = \underset{v_j \in V}{argmax} P(v_j) \prod_i P(MS_i/v_j) \quad (6)$$

Here, $V = \{c_{d+}, c_{d-}\}$, and MS_i is the basic service of solution s. We use the normalized P to represent the prior value of solution s.

As an example, Table 4 shows a segment of historical data. Based on the above formulas, the posterior probability of a

certain candidate service and solution can be calculated, specifically, $P(c_1/s_{11}) = \frac{\frac{4}{6}\times\frac{3}{4}}{\frac{4}{6}} = 0.75$, $P(R1/c_{1+}) = \frac{4}{6}\times\frac{3}{4}\times\frac{4}{6}\times\frac{1}{2}\times\frac{1}{2} = \frac{1}{12}$, and $P(R1/c_{1-}) = \frac{1}{144}$.

Table 4 A segment of historical data

No.	Request cluster	Service execution record	User satisfaction
R1	c_1	$s_{11}, s_{22}, s_{31}, s_{42}$	c_{1+}
R2		$s_{12}, s_{21}, s_{32}, s_{41}$	
R3		$s_{11}, s_{21}, s_{33}, s_{41}$	
R4		$s_{11}, s_{22}, s_{32}, s_{44}$	
R5		$s_{11}, s_{22}, s_{31}, s_{42}$	c_{1-}
R6		$s_{12}, s_{21}, s_{32}, s_{41}$	
R7	c_2	$s_{16}, s_{24}, s_{33}, s_{44}, s_{51}$	c_{2+}
R8		$s_{16}, s_{23}, s_{32}, s_{44}, s_{53}$	
R9		$s_{11}, s_{21}, s_{32}, s_{42}, s_{53}$	
R10		$s_{11}, s_{21}, s_{32}, s_{42}, s_{53}$	c_{2-}
R11	c_3	$s_{12}, s_{22}, s_{34}, s_{44}, s_{54}, s_{63}, s_{77}$	c_{3+}
R12		$s_{14}, s_{24}, s_{31}, s_{43}, s_{51}, s_{62}, s_{71}$	
R13		$s_{11}, s_{22}, s_{32}, s_{43}, s_{52}, s_{62}, s_{75}$	
R14		$s_{14}, s_{24}, s_{31}, s_{43}, s_{51}, s_{62}, s_{71}$	c_{3-}

Definition 2 (Correlation). The correlation is an evaluation value of the association relationship of services $(s_{ik}, s_{jm}, \dots)$ that are synergistically used to meet the request in c_d. The correlation between services includes business, statistical and QoS correlations. In this paper, we focus on both the business and QoS correlations. For the business correlation, the correlation value of synergistic services can be calculated as follows:

$$C_b\left(c_d/\left(MS_{ik}, MS_{jm}, \dots\right)\right) = \begin{cases} \frac{N^+ - N^-}{N^+}, \text{if } N^+ - N^- > 0 \\ 0, \quad else \end{cases} \quad (7)$$

where N^+ and N^- are the occurrence numbers of $MS_{ik}, MS_{jm}, \dots$ in positive and negative records, respectively. Take the synergistic two MSs s_{21} and s_{41} in Table 3 as an example, $C_b\left(c_1/\left(MS_{2,1}, MS_{4,1}\right)\right) = \frac{2-1}{2} = \frac{1}{2}$.

For the QoS correlation, we only consider the two neighbour services, which have higher QoS dependence. In the historical data of the request class c_d, except for the starting and final MSs of each record, each MS has two sets of neighbouring MSs. For a given MS_{ik}, the first set F includes MSs that executed before MS_{ik} in different records, and the second set S contains MSs that executed after MS_{ik}. For example, in Table 2, two sets of neighbouring MSs of s_{21} in c_1 are $\{s_{11}, s_{12}\}$ and $\{s_{32}, s_{33}\}$, respectively. For each set, there is an average value of the QoS correlation values for evaluating the difference degree of the QoS correlation between MS_{ik} and each MS_j in the set. The average value can be calculated by

$$\varphi_{QoS}\left(c_d/\left(MS_{ik}, MS_j\right)\right) = \frac{\sum_{j=1}^{m} C_{QoS}\left(c_d/\left(MS_{ik}, MS_j\right)\right)}{m} \quad (8)$$

where $C_{QoS}\left(c_d/\left(MS_{ik}, MS_j\right)\right)$ is the normalization function and m is the number of MSs in the set. Its formula is defined as the following:

$$C_{QoS}\left(c_d/\left(MS_{ik}, MS_j\right)\right) = \frac{f\left(MS_{ik}, MS_j\right) - f_{min}\left(MS_{ik}, MS_j\right)}{f_{max}\left(MS_{ik}, MS_j\right) - f_{min}\left(MS_{ik}, MS_j\right)} \quad (9)$$

where $f\left(MS_{ik}, MS_{jm}\right)$ is the utility function, being the same as the function in Eq. 4 for calculating an average aggregated QoS value of two neighbour services in the historical data. Its formula is defined as the following:

$$f\left(MS_{ik}, MS_{jm}\right) = \begin{cases} \frac{\sum_{k=1}^{N^+}\sum_{j=1}^{n} w_j q_j - \sum_{k=1}^{N^-}\sum_{j=1}^{n} w_j q_j}{N^+ - N^-} & if\ N^+ \geq N^- \\ 0 & else \end{cases} \quad (10)$$

where N^+ and N^- are the occurrence numbers of MS_{ik} and MS_{jm} in positive and negative records, respectively; q_j is the aggregated value of the jth QoS criterion for the two neighbouring services MS_{ik} and MS_{jm} and should be normalized by Eq. 2 or Eq. 3 based on all aggregated jth QoS values of MS_{ik} and each MS_{jm} in a neighbouring service set.

When $\varphi_{QoS}\left(c_d/\left(MS_{ik}, MS_j\right)\right)$ is very small, the difference degree of the QoS correlation between MS_{ik} and each MS_j is very large. Hence, services with the largest value of $C_{QoS}\left(c_d/\left(MS_{ik}, MS_j\right)\right)$ should be considered services with the highest QoS correlation.

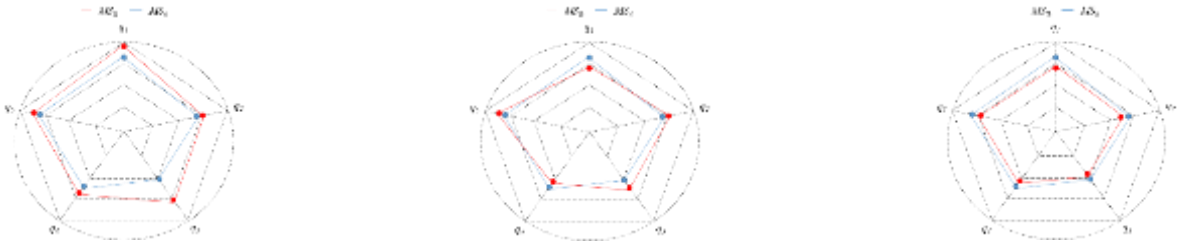
For example, we assume that we initially use $f\left(MS_{ik}, MS_{jm}\right)$ to calculate two values {2, 1} for services in $\{s_{11}, s_{12}\}$ and then use $C_{QoS}\left(c_d/\left(MS_{ik}, MS_j\right)\right)$ to obtain two normalized values {1, 0}. The difference degree of QoS correlation between s_{21} and s_{11} and between s_{21} and s_{12} can be calculated by $\varphi_{QoS}\left(c_d/\left(MS_{ik}, MS_j\right)\right) = 0.5$. In other words, MSs s_{21} and s_{11} have a higher QoS correlation than do s_{21} and s_{12}, and the greater value with $C_{QoS}\left(c_d/\left(MS_{ik}, MS_j\right)\right)$ represents the higher QoS correlation.

Definition 3 (Similarity). The similarity is the similarity degree of QoS values between two candidate MSs that have the same service functionalities and belong to the same candidate service set. There are three types of similarity. As shown in Fig. 2, the first type is called superior similarity, which means that each QoS index $MS_{ik} \cdot q_j$ is better than $MS_{it} \cdot q_j$ (we assume that all QoS attributes are positive in Fig. 2). The second type is the general similarity, which represents that some values of $MS_{ik} \cdot q_j$ are better than the corresponding value of $MS_{it} \cdot q_j$, and the remaining values of the QoS attributes are not better. The last type is the opposite of the first type, namely, the inferior similarity. In this paper, we only consider the superior and relative similarities. Moreover, for new services with unknown QoS values, we employ prediction methods (Karim et al. 2015; Feng and Huang, 2018) to predict QoS values of new services for calculating similarities. The similarity values can be calculated by Eq. 12 and Eq. 13.

$$S_s(c_d/MS_{ik}, MS_{it}) = 1 + \frac{|f(MS_{it}) - f(MS_{ik})|}{f(MS_{ik})} \quad (11)$$

$$S_r(c_d/MS_{ik}, MS_{it}) = \frac{1}{\sqrt{\sum_j (MS_{it} \cdot q_j - MS_{it} \cdot q_j)^2 + 1}} \quad (12)$$

Here, $f(X) = \sum_{j=1}^{n} w_j q_{j,X}$ is the utility function of the QoS value for an MS X; $q_{j,X}$ is the value of a certain QoS attribute normalized by Eq. 2 or Eq. 3 based on the same type of QoS values of MSs in the same candidate MSs set; S_r is the general similarity function based on the Euclidean distance; and $MS_{it} \cdot q_j$ is the normal value of QoS.



(a) Superior similarity (b) General similarity (c) Inferior similarity

Fig. 2 Examples of different similarities

With the above-defined SDFs, for a specific service request in c_d with the historical data about c_d, the prior feature can be used to identify prior services and solutions that can be used to satisfy the request. The correlation feature can further find the high-frequency service schemas from the perspectives of business correlation and QoS correlation. These schemas are additional supplements for discovered high-probability services. Moreover, based on the identified services and schemas, using the similarity feature to search similar services with them, the number of feasible services can increase, and additional outstanding services can be found based on the superior similarity. In other words, SDFs can be used to effectively segment the whole service space and reduce the search space for algorithms to find the global optimal solution. With the increasing candidate service sets, services identified by prior and correlation features can be used to compose the local optimal solutions, and services found based on similarity can be used to compose the global optimal solution. The boundaries of the space division are clearer. Consequently, the reasonable utilization of SDFs to divide the whole candidate service set and combination to design appropriate search strategies of algorithms can dramatically promote the performance of algorithms for searching for the global optimal solution. Next, we define the division of service space and discuss the design strategies of algorithms influenced by SDFs.

4.2 Service space division

With fine-grained features, the whole candidate service set for a service request S_p can be divided into the prior Service Set (P*riS*), Correlation Service Set (C*orS*), Similar Service Set (S*imS*) and General Service Set (G*enS*). Accordingly, the service space is divided into four subspaces. The formal description of these service sets can be expressed as follows.

1. $PriS(S_p, c_d)=\{s|\ s\in S_p \wedge P(c_{d+}/s)\geq P_t\}$

where $Sp = jS_j$ is the whole candidate service set for a service request in c_d, S_j is the candidate service set for each subtask in the service process of c_d, P_t is the threshold of the prior value of the candidate service based on the partition function $U(h) \geq T$, and P_t= 0.5 might be the lower limit value with a certain value T.

2. $CorS(S_p, c_d)= CorS(S_p, c_d)_1 \vee CorS(S_p, c_d)_2$

$CorS(S_p, c_d)_1=\{(s_i, s_j\ldots)\ |\ s_i, s_j\ldots \in S_p, \exists\ C_b(c_d/(s_i, s_j, \ldots)) \geq C_{b0}\}$

$CorS(S_p, c_d)_2=\{(s_i, s_j)|\ s_i\in PriS(S_p, c_d) \wedge s_j\in S_p, \exists\ C_{QoS}(c_d/(s_i, s_j)) \geq C_{QoS1} \wedge \varphi_{QoS}(c_d/(s_i, s_j)) \leq k\}$

where C_{b0} and C_{QoS1} are thresholds based on the partition function $U(h) \geq T$; C_{b0}= 0 might be the lowest limits with T; and $C_{QoS1} = 1$ must be the upper limits when $\varphi_{QoS}(c_d/(s_i, s_j))$ is smaller than the threshold k; k should be smaller than 0.5.

3. $SimS(S_p, c_d)=\{s|\ s\in S_p \wedge s_j\in (PriS(S_p, c_d) \vee CorS(S_p, c_d)), \exists S_s(c_d/s, s_j) \geq S_{s0} \vee S_r(c_d/s, s_j) \geq S_{r0}\}$

where S_{s0} and S_{r0} are thresholds of similarity.

4. $GenS(S_p, c_d)= S_p- PriS(S_p, c_d)- CorS(S_p, c_d)- SimS(S_p, c_d)$

The above descriptions indicate dependencies among the four sets. $CorS(S_p, c_d)$ partly relies on $PriS(S_p, c_d)$ because the subset $CorS(S_p, c_d)_2$ only covers services that have a QoS correlation with services in $PriS(S_p, c_d)$. If a service belongs to both $PriS(S_p, c_d)$ and $CorS(S_p, c_d)$, it should be allocated to $CorS(S_p, c_d)$ to retain the correlation between services and ensure that $P_s \Lambda C_s = \emptyset$. $SimS(S_p, c_d)$ is completely dependent upon $PriS(S_p, c_d)$ and $CorS(S_p, c_d)$. It attempts to cover all similar and superior services with services in $PriS(S_p, c_d)$ and $CorS(S_p, c_d)$. In this case, if the $PriS(S_p, c_d)$ contains feasible services for a certain request, based on the dependence relationships, $CorS(S_p, c_d)$ and $SimS(S_p, c_d)$ then can provide better services (even new services that are not being used and records in historical data can be identified by the similarity feature) for this request. It means that the first three feature sets might contain the global optimal solution. The procedure for dividing the service space is shown in Alg. 1.

Alg. 1 Processing the service space

input: Historical data *H*, *U(h)*, *T,* candidate service set *Sp*
Output: four candidate service sets
PriS, *CorS*, *SimS* and *GenS*← empty set
H^+, H^- ←employ *U(h)*≥*T* to divide *H*
Loop for *i* **from** 1 **to** |*Sp*| **do**
  P*i*← employ Eq. (5) to calculate prior value for service *i*
  If P*i*≥P*t*=0.5 **then** add service *i* to *PriS* and add P*i* to value set *Vp*
Loop for *i* **from** 1 **to** number of solutions in *PriS* **do**
  P*s*← employ Eq. (6) to calculate prior value for solution *i*
  Add P*s* to *Vp* and rank solution with P*s*
C*s*← employ Apriori algorithm to get service schemes
Loop for *i* **from** 1 **to** |C*s*| **do**
  C*i*← employ Eq. (8) to calculate correlation value for service scheme *i*
  If C*i* > C_{b0}=0 **then** add service scheme *i* to *CorS* and C*i* to value set *Vc*
PriS← *PriS* - (*PriS* Λ *CorS*)
Loop for *i* **from** 1 **to** |*PriS*| **do**
  F, *S*← construct two neighboring service sets for service *i*
  If |*F*| ≥ 2 **then** *φ*← employ Eq. (9) to calculate the degree value
  If *φ*≤ *k*=0.5 **then** add service *i* and the service in *S* that has the value 1 calculated by Eq. (10) to *CorS* (if service *i* and the service are not in *CorS*) and add the corresponding value 1 to *Vc*
  If |*S*| ≥ 2 **then** *φ*← employ Eq.(9) to calculate the degree value
  If *φ*≤ *k*=0.5 **then** add service *i* and the service in *S* that has the value 1 calculated by Eq.(10) to *CorS* (if service *i* and the service are not in *CorS*) and add the corresponding value 1 to *Vc*
PriS← *PriS* - (*PriS* Λ *CorS*)

```
GenS← Sp-(PriS Λ CorS)
Loop for i from 1 to |PriS Λ CorS| do
  S← GenS Λ Sj
  loop for j from 1 to |S| do
  If service j is the superior service then Sj← employ Eq. (12)
        to calculate the similarity value, add service j to
        SimS, and add value Sj to Vs
  If service j is the relative service then do
    Sj← employ Eq. (13) to calculate the similarity value
    If Sj≥ Sro=0.90 then add service j to SimS, add value Sj to
          Vs
GenS← GenS-(GenS Λ SimS)
Return PriS, CorS, SimS and GenS
```

The above algorithm can be used to divide the service space properly. The thresholds for P*riS* and C*orS* are set with P*t* = 0.5 and C_{b0} = 0. They rely on the threshold *T*. When a more strict value *T* (a larger one if *T* is positive or otherwise a small one) is used, P*t* and C_{b0} should be set with smaller value, whereas greater values might be appropriate. The threshold *k* is set to 0.5, which is a basic value for the difference degree of QoS correlations. If prominent QoS correlations are expected, a small value should be used. Similarly, the threshold S_{s0} is set to 1, which means that if a service is the superior service over a service in P*riS* and C*orS*, it should be classified in the S*imS*. The relative similarity threshold S_{ro} is set to 0.9. It is a key value influencing the scale of S*imS*. Except for the thresholds S_{s0} and S_{ro}, the scale of S*imS* also relies on the scales of P*riS* and C*orS*. In other words, neglecting the influence of the thresholds of P*riS,* C*orS* and S*imS*, the utility function and threshold *T* determine the scales of the three feature sets. Once four service sets are determined, the CMfg system can maintain and update these service sets regularly, which can be used to facilitate the solving of SCOS problems.

Considering the superiority of divisions of service space, the strategies for improving algorithms can be summarized as follows:

(1) Initialize the appropriate number of individuals in the initial population from the first three service sets to obtain better initial positions;

(2) Control the search direction of the algorithms for widely searching the three feature sets first; and

(3) Improve the fitness function to make more individuals from the three feature sets survive and to reduce deception problems.

5 Service domain feature-oriented genetic algorithm

Based on the strategies for improving algorithms fully considering the influence of service domain features, this paper proposes a service domain feature-oriented genetic algorithm (SDF-GA) based on improving the initialization of the initial population, genetic operators and fitness function.

5.1 Initial population of SDF-GA

The searching strategy used by SDF-GA is to search for the global optimal solution in all subspaces simultaneously, which means that individuals in the initial population should be initialized from four service sets. To retain the diversity of the population, we initialize better individuals and deeply search each subspace, and a certain number of individuals should be generated from each set. The number of individuals from each feature set can be equal or correspond to the proportion of feature sets. In this paper, the proportion of each set is used as the number of individuals generated from each set. All individuals can be expressed as $X=(x_1, x_2, \ldots, x_n)$, where x_n is the selected service for a subtask. As shown in Fig. 3, an individual is encoded by the real serial numbers of the selected service, and the length of an individual is the number of subtasks involved in an MSC.

Individuals from G*enS* are randomly generated. For the P*riS*, C*orS* and S*imS* sets, three strategies are used to initialize individuals. The first strategy is to randomly generate individuals. The second strategy is to generate individuals for max $\prod (1+v_j)$, and v_j is the feature value of selected service *j*. The third strategy is to reuse the existing solution, which has the highest probability with feature values based on Eq. 6. Use probabilities of these strategies are equal. Alg. 2 shows the common procedure for

generating individuals from three feature sets. When the algorithm generates individuals from C*orS*, different sets of correlation services are selected randomly simultaneously.

Alg 2. Generate individuals from the feature set

input: service set S, feature value set V, N
Output: a set of individuals
Individual set I_s← empty set
Loop for i **from** 1 **to** N **do**
x ← a random integer from 1 to 3

switch (x)
  case 1:
    Xi← randomly select a service x_n from S for each subtask to generate an individual
    add Xi to I_s, **break**
  case 2:
    Xi← randomly select a service x_n from S for each subtask to generate an individual with max $\prod(1+v_j)$
    add Xi to I_s, **break**
  case 3:
  If (solutions are insufficient) **then do**
    k ←a random integer from 1 to 2
    If k=1 **then** Xi← randomly select a service x_n from S for each subtask to generate an individual
    else Xi← randomly select a service x_n from S for each subtask to generate an individual with max $\prod(1+v_j)$
  else Xi← select the best solution from S
  add Xi to I_s, **break**
return I_s

The above algorithm is a universal algorithm for generating individuals from three feature sets. When a feature set lacks services for constituting integrated individuals, relative services can be selected from other feature sets in the corresponding candidate service list first; then, the last choice is to select a service from G*enS*. The feature values of all services in G*enS* are set to 0.

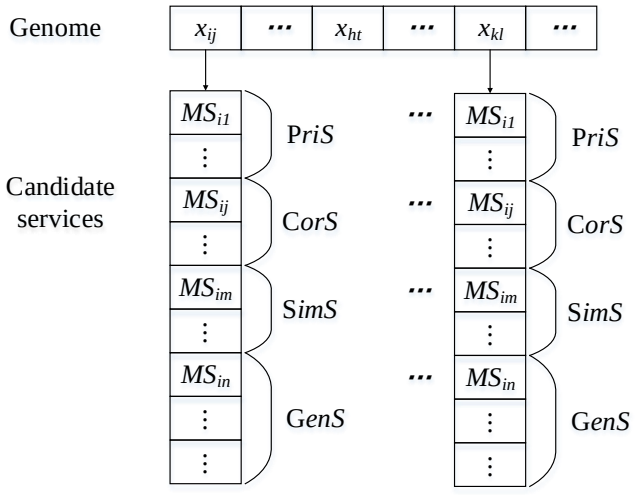


Fig. 3 The coding scheme of a genome

5.2 Operators of SDF-GA

The crossover operator in SDF-GA is a single-point crossover that selects a location of two individuals to exchange their

services after the location is selected. To search the three feature sets as widely as possible, three strategies are used to generate one offspring based on two randomly selected parent individuals. The first strategy is to search the location of two parent individuals for crossover to generate one offspring with the maximum value of $\prod (1+v_j)$. The second strategy is to find the location of two parent individuals for crossover to generate one offspring with a maximum value of $\sum N_j$, and Nj is the number of services from the feature set. The third strategy is to randomly select a location for crossover to generate an offspring with random maximum $\prod (1+v_j)$ or $\sum N_j$. The probabilities of being used for each strategy are equal. Alg. 3 shows the reproduction algorithm with two selected parents. Two parents generate one offspring in this algorithm.

Alg. 3 REPRODUCTION of an individual

Input: X*i*, X*j* feature value sets V*p* V*c* V*s*
Output: an individual
S*o* ← empty set
X*n*, V*n* ← empty individual and zero
x ← a random integer from 1 to 3
Switch (*x*)
  Case 1:
    loop for *i* **from** 1 to $|X_i|$-1 **do**
      S*o* ← generate two offsprings by exchanging services X*i* and X*j* after service *i*
      Loop for *i* **from** 1 to 2 **do**
      V← calculate feature value with $\prod (1+v_j)$ for offspring *i* in S*o*
    If V>V*n* **then** V*n* ← V and X*n* ←offspring *i*
  break
  Case 2:
    loop for *i* **from** 1 to $|X_i|$-1 **do**
      S*o* ← generate two offsprings by exchanging services X*i* and X*j* after service *i*
      Loop for *i* **from** 1 to 2 **do**
      V← calculate the number of services in feature sets with $\sum N_j$ for offspring *i* in S*o*
    If V>V*n* **then** V*n* ← V and X*n* ←offspring *i*
  break
  Case 3:
    k ← Randomly select a number from 1 to |Xi|
    S*o* ← generate two offsprings by exchanging services X*i* and X*j* after service *k*
    t← Randomly select a number from 1 to 2
    If *t*=1 **then do**
      Loop for *i* **from** 1 to 2 **do**
      V← calculate feature value with $\prod (1+v_j)$ for offspring *i* in S*o*
    If V>V*n* **then do**
      V*n* ← V and X*n*← offspring *i*
  Else do
      Loop for *i* **from** 1 to 2 **do**
      V← calculate the number of services in feature sets with $\sum N_j$ for offspring *i* in S*o*
    If V>V*n* **then do**
    V*n* ← V and X*n* ←offspring *i*
  break
Return X*n*

The mutation operator is also a single-point mutation that selects a position of an individual and exchanges the service in the selected position with the other service in the candidate service list. To increase the diversity of new generations, a simple strategy is used to mutate the selected individual. The mutation operator randomly selects a position in the newly generated individual and randomly replaces the selected candidate service with another one from other sets in the candidate list. Specifically, if the selected service belongs to the feature sets, then the substitutive service should be selected from *GenS*. Otherwise, if the selected service is from G*enS*, the substitutive service is equally selected from the feature sets. Alg. 4 shows the algorithm for the mutation operator.

Alg. 4 MUTATE

Input: Xi
Output: an individual
$k \leftarrow$ Randomly select a number from 1 to |Xi|
If x_k is from G*ens* **then** do
  $x \leftarrow$ a random integer from 1 to 3
  Switch (x)
    Case 1:
      $x_k \leftarrow$ Exchange x_k with randomly selected service from P*riS* **break**
    Case 2:
      $x_k \leftarrow$ Exchange x_k with randomly selected service from Co*rS* **break**
    Case 3:
      $x_k \leftarrow$ Exchange x_k with randomly selected service from S*imS* **break**
Else $x_k \leftarrow$ Exchange x_k with randomly the selected service from G*enS*
Return X*i*

5.3 Fitness function of SDF-GA

For SDF-GA, the fitness of an individual should not only reflect its QoS utility but also indicate how the individual can satisfy SDF constraints for searching feature subspaces. Consequently, the optimization problem and SFD constraints should all be considered. For SDF-GA, the fitness function can be defined as follows:

$$F(X_i) = w_1\varphi\big(f(X)\big) + w_2\frac{\eta}{T}f_{SDF}(X_i) \quad (13)$$

where $f(X_i)$ is the objective function and $\varphi(y)$ is the transfer function based on transformational rules from the objective to the fitness functions. $f_{SDF}(X_k) = \prod_j v_j$ is the SDF constraint function, v_j is the feature value of service j in the individual X_k, and N_j is the service number from each feature set. $0 \le \eta \le \frac{maxf(X_k)-minf(X_k)}{maxf_{SDF}(X_k)}$ is the balanced coefficient for the SDF function T is the number of iterations, and w_i is the weight, $\sum_i w_i = 1$. The coefficientη/T decreases with each iteration, which ensures that later iterations focus on the true fitness of individuals and prevent the algorithm from becoming trapped in local optima in a certain subspace. In this paper, we set to w_1=w_2=0.5.

However, the divided service space based on SDFs makes the search landscape more rugged. In searching for the global optimal solution, feature subspaces that might contain the global optimal solution must be fully explored. The fitness-sharing niching technology (Wu et al. 2014) is used by the GA to perform more searches in each subspace and to maintain the diversity of a population to mitigate premature convergence. The amendatory fitness function is defined as follows:

$$F_{nic}(X_i) = \frac{F(X_i)}{\sum_{h\in population} S_{h,i}} \quad (14)$$

Here, $\sum_{h\in population} S_{h,i}$ represents the crowdedness degree of individual X_i in its niche. $S_{h,i}$ is the sharing value between X_i and X_h and can be calculated as follows:

$$S_{h,i} = \begin{cases} \frac{\sigma - d_{h,i}}{\sigma}, if d_{h,i} < \sigma \\ 0 \quad , if d_{h,i} \ge \sigma \end{cases} \quad (15)$$

where σ is a defined sharing radius and $d_{h,i}$ is the number of different services between X_h and X_i.

Using the amendatory fitness SDF-GA, highly similar individuals are discouraged. For example, X_i= {x_{12}, x_{23}, x_{37}, x_{42}, x_{51}, x_{66}}, X_j={x_{16}, x_{29}, x_{37}, x_{42}, x_{55}, x_{62}} and X_k={x_{16}, x_{29}, x_{37}, x_{43}, x_{55}, x_{62}} are three individuals with fitness values of 0.6, 0.8 and 0.9, respectively. $d_{i,j}$=4, $d_{i,k}$=5 and $d_{j,k}$=1; if σ is set to 2, then $S_{i,j}$=0, $S_{i,k}$=0, $S_{j,k}$=0.5 and $S_{i,i}$= $S_{j,j}$= $S_{k,k}$=1. Using Eq. 14, $F_{nic}(X_k)$=$F(X_k)$/($S_{i,k}$+ $S_{j,k}$+ $S_{k,k}$)=0.9/1.5=0.6. Similarly, we find $F_{nic}(X_i)$=0.6 and $F_{nic}(X_j)$=0.533. Hence, the two closer individuals X_j and X_k are suppressed to some extent, and individual Xi, which uniquely exploits areas of the search space, is encouraged for evolution.

Based on the above-designed strategies, Alg. 5 shows the pseudocode of SDF-GA.

Alg. 5 Pseudocode of SDF-GA

Input: N, r, m, A, C, S_1, S_2, w_i
Output: an individual
preprocessing the service space: employ Alg. 1 to generate feature sets
initialize initial population P: employ Alg. 2 to generate $\alpha * N$, $\beta * N$, and $\gamma * N$ individuals from each feature set and randomly generate $\delta * N$ individuals from G*enS*
fitness evaluation: for each individual, employ Eq. 13 to calculate the fitness $F_{nic}(p_i)$
repeat
  new-population Pn ← empty set
  selection: add $(1-r) * N$ individuals from P to P_n with the roulette method
  crossover and mutation:
  loop for i **from** 1 to ∞ **do**
  newchild ← empty
  x, y ← randomly select two individuals from P
  child ← employ Alg. 3 to generate an individual
  $\Delta f = f(child) - max(f(x), f(y))$
  if Δf>0 **Then** *newchild* ← *child*
  else *newchild* ← *child* only accepted with probability 0.5
  If *newchild* is not empty **then do**
    n++
    If small random probability $s \leq m$ **then do**
      newchild ←employ Alg. 4 to generate an individual
    Add *newchild* to *Pn*
  If $n = r * N$ **then break**
  update: P ← *Pn*
until stopping criteria are satisfied
return the best individual in P, according to $f(p_i)$

5.4 Convergence analysis

Intuitively, SDF-GA can achieve better convergence than traditional intelligence optimization algorithms, such as T-GA, PSO, and ABC, because it employs rich prior knowledge. We present an extreme example – if SDF-GA directly uses solutions found by a traditional intelligent optimization algorithm as the initial population to search for the best solution, SDF-GA will not perform poorly. Considering that the most distinguishing characteristic of SDF-GA with traditional intelligent optimization algorithms is the use of prior knowledge, we use the Bayes theorem to briefly prove the superiority of SDF-GA and designate T-GA as an exemplar of traditional intelligent optimization algorithms to contrastively analyse the performance boundaries of SDF-GA. A lemma for T-GA and a property of SDF-GA are introduced first.

Let R be the search space of a manufacturing cloud service composition, R={r_1, r_2, … r_h} be a set of subspaces, $i = 1 hr_i = R$. Let $y_b = f$ be the best solution, B={$y_n | \forall y_n \in R \wedge \vee\, y_b - y_n \vee \leq \varepsilon$} be a set of feasible solutions, and ε be an arbitrarily small number. Each subspace has at most one set of feasible solutions B. The subspace that contains B is called the optimal subspace. Let $A^{(t)}$ ={$y_i | \forall y_i \in R$} be the population of T-GA/SDF-GA of generation t and let $P\left(A^{(t)}\right)$ be the probability of finding a solution in B. In other words, it is the probability that individuals of $A^{(t)}$ include elements of B. The set R={r_1, r_2, … r_h} can be further divided into two subsets. One subset includes subspaces with elements of B; the other set contains subspaces that do not include include elements of B. Let u be the number of the first set; then, the number of the second set is h-u. We have $P\left(A^{(t)}\right) = \sum_{j=1}^{u} P\left(A_j^{(t)}\right)$, and $P\left(A_j^{(t)}\right)$ is the probability that individuals from $A^{(t)}$ located in the optimal subspace j contain elements of B. When the algorithm has not converged at iteration t, we also can have $\sum_{i=1}^{h-u} P\left(A_i^{(t)}\right) + \sum_{j=1}^{u} P\left(A_j^{(t)}\right) = 1$. When the algorithm converges in t iterations, we have $\sum_{j=1}^{u} P\left(A_j^{(t)}\right) = 1$, $\sum_{i=1}^{h-u} P\left(A_i^{(t)}\right) = 0$ (Jiang et al. 2014).

(Lemma 1). Let $A^{(0)}$ and $P\left(A^{(0)}\right)$ be the initial population and its probability for containing the element in B. Let $\Delta p = \frac{\sum_{i=1}^{t} P\left(A^{(t)}\right) - P\left(A^{(t-1)}\right)}{t}$ be the average difference of probability between adjacent iterations. The convergence time of T-GA is

positively correlated with $P\left(A^{(0)}\right)$ and negatively correlated with Δp:

$$E(t) = \frac{1-P\left(A^{(0)}\right)}{\Delta p} \quad (16)$$

According to lemma 1, we can conclude that a better initial population can accelerate the convergence velocity of GA and its derived algorithms. Simultaneously, better strategies for crossover and mutation that lead to a greater Δp can also improve the convergence speed without considering the time consumed by the strategy implementation.

Consequently, if SDF-GA has a better initial population or strategies for crossover and mutation, SDF-GA can achieve better performance. Indeed, SDF-GA fully utilizes the prior knowledge about the distribution of *B* in the subspaces. We need to learn the property of SDF-GA before we discuss the superiority of SDF-GA. We assume that the prior knowledge used by SDF-GA is from the final solutions of T-GA for the same SCOS problem. From the perspective of Bayesian theory, the probability of the initial population of SDF-GA is a posterior probability of the results of T-GA about the distribution of *B* in the subspaces. Consequently, one property of SDF-GA can be defined as follows.

(Property 1). Let $A_{SDFs-GA}^{(0)}$ be the initial population of SDF-GA. Let $P_{SDFs-GA}\left(A_j^{(0)}\right)$ be the probability that individuals in $A_{SDFs-GA}^{(0)}$ located in the optimal subspace *j* contain elements of *B*. Similarly, let $A_{T-GA}^{(P)}$ be the available solutions of T-GA, and let $P_{T-GA}\left(A_j^{(P)}\right)$ be the prior probability that individuals from $A_{T-GA}^{(P)}$ located in subspace *j* contain elements in *B*. Let $P_{T-GA}\left(B/A_j^{(P)}\right)$ be the occurrence probability that elements of *B* are located in subspace *j*. The $P_{SDFs-GA}\left(A_j^{(0)}\right)$ is a posterior probability for the distribution of *B* in each subspace:

$$P_{SDF-GA}\left(A_j^{(0)}\right) = P_{T-GA}\left(A_j^{(P)}/B\right) = \frac{P_{T-GA}\left(A_j^{(P)}\right)P_{T-GA}\left(B/A_j^{(P)}\right)}{\sum_h P_{T-GA}\left(A_j^{(P)}\right)P_{T-GA}\left(B/A_j^{(P)}\right)} \quad (17)$$

where $P_{T-GA}\left(B/A_j^{(P)}\right) = \frac{k_j}{\sum_{j=1}^{h} k_j}$, k_j is the number of best solutions in the subspace *j* found by T-GA in each iteration, and $\sum_{j=1}^{h} P_{T-GA}\left(B/A_j^{(P)}\right) = 1$.

With this property, we can have two lemmas that can be used to confirm the superiority of SDF-GA.

(Lemma 2). Let $A_{T-GA}^{(0)}$ be the generated feasible individuals of T-GA, and let $A_{SDF-GA}^{(0)}$ be the initial population of SDF-GA based on the prior knowledge abstracted from $A_{T-GA}^{(0)}$. Let $P_{SDF-GA}\left(A^{(0)}\right)$ be the probability that individuals of $A_{SDF-GA}^{(0)}$ include elements of *B*. In other words, $P_{SDF-GA}\left(A^{(0)}\right)$ is the probability that individuals of $A_{SDF-GA}^{(0)}$ are initialized in the optimal subspaces. Let $P_{T-GA}\left(A^{(0)}\right)$ be the probability that individuals of $A_{T-GA}^{(0)}$ include elements of *B*. Then, we can have $P_{SDF-GA}\left(A^{(0)}\right) \geq P_{T-GA}\left(A^{(0)}\right)$, which implies that SDF-GA can initialize a better initial population than can T-GA.

Based on the proof of lemma 2 (see the Appendix), we can conclude that when *u*=*h* or $P_{T-GA}\left(B/A_i^{(0)}\right) = P_{T-GA}\left(B/A_j^{(0)}\right) = \frac{1}{h}$, $P_{SDF-GA}\left(A^{(0)}\right)$=$P_{T-GA}\left(A^{(0)}\right)$. Combining with lemma 1, this conclusion indicates that when all four subspaces are the optimal subspace due to the unreasonable division of the service space or the prior knowledge of $P_{T-GA}\left(B/A_j^{(0)}\right)$, SDF-GA can achieve the same performance as T-GA. This indication not only demonstrates the superiority of SDF-GA because it can perform better than T-GA, in which case the feature sets are an optimal subspace based on the sufficient prior knowledge, but also provides the lower boundary of the performance of SDF-GA. When all four sets are the optimal subspace, the poorest performance of SDF-GA is the same as the performance of T-GA.

Similarly, when *u*=0, SDF-GA can achieve the same performance as T-GA, which implies that when prior knowledge is insufficient and the division of the service space is unreasonable, the optimal subspaces cannot be identified, and SDF-GA only performs similarly to T-GA, which randomly searches the whole service space.

(Lemma 3). Let $A^{(0)}$ be the same initial population of T-GA, SDF-GA, $A_{T-GA}^{(1)}$ be the first generation of T-GA and $P_{T-GA}\left(A^{(1)}\right)$ be the probability that individuals of $A_{T-GA}^{(1)}$ include elements of *B*. Let $A_{SDF-GA}^{(1)}$ be the first generation of SDF-GA, and let $P_{SDF-GA}\left(A^{(1)}\right)$ be the probability that individuals of $A_{SDF-GA}^{(1)}$ include elements of *B*. We assume that the operators of SDF-GA are based on feature sets abstracted from historical data of $A_{T-GA}^{(1)}$. Then, we can have $P_{SDF-GA}\left(A^{(1)}\right) \geq P_{T-GA}\left(A^{(1)}\right)$. For the proof, please see the Appendix.

Lemma 3 indicates that with the same initial population and prior knowledge, SDF-GA has a better convergence performance based on the inference $\Delta p_{SDFs-GA} = P_{SDF-GA}\left(A^{(1)}\right) - P\left(A^{(0)}\right) \geq \Delta p_{T-GA} = P_{T-GA}\left(A^{(1)}\right) - P\left(A^{(0)}\right)$. However, it also implies that SDF-GA can find more-optimal solutions than can T-GA.

Based on lemmas 1-3, we can have two corollaries.

Corollary 1. For a problem of SCOS in CMfg, if the division of the service space is reasonable, when SDF-GA and T-GA can both find the best solution, SDF-GA might have a better convergence performance than T-GA.

Corollary 2. For a problem of SCOS in CMfg, if the division of the service space is reasonable, and SDF-GA and T-GA cannot both find the best solution, then SDF-GA might achieve not only a better performance in searching for the best solution but also better convergence performance than T-GA.

6 Case study

6.1 Objectives and dataset

The performance of SDF-GA is evaluated using three metrics: convergence speed, capability of finding the best solution with limited iterations and probability of obtaining the global optimum solution. Different objectives are considered in analysing SDF-GA:

1) Analysing the effects of the SDFs on the performance of the proposed algorithm;
2) Assessing the effectiveness of the optimization strategies for the proposed algorithm; and
3) Comparatively evaluating the performance of SDF-GA against other SDF-oriented algorithms.

Consequently, three case studies were implemented using Visual Studio 2010 with the C# language on a PC with an Intel(R) i5-6200U @ 2.30 GHz CPU and 8 GB of memory. One dataset is based on a real-world dataset from a private CMfg. This CMfg belongs to one of the largest manufacturers in China, who is trying to take advantage of CMfg to realize industry 4.0. Based on this CMfg, a specific demand in the customized production of household electrical appliances can be easily satisfied by finding an optimal solution based on the service process with 72 subtasks (see Fig. 4; black cuboids represent a set of the same type of subtasks). For each subtask, there are 58 qualified candidate services (including services provided by SME) on average, and each service has five QoS parameters: Cost (*C*), Execution time (*Et*), Response time (*Rt*), Reliability (*R*), and Availability (*A*). According to the customer requirement, we can assume that minimizing the Execution time Min $f\left(Et_{n,1}, \ldots, Et_{n,j}\right)$ is the target for the manufacturing cloud service composition with QoS constraints $\varphi\left(C_{k,1}, \ldots, C_{k,j}\right)$≤6000, $\varphi\left(Rt_{k,1}, \ldots, Rt_{k,j}\right)$≤50*h*(hours), $\varphi\left(Et_{k,1}, \ldots, Et_{k,j}\right)$≤*17d*(days), $\varphi\left(R_{k,1}, \ldots, R_{k,j}\right)$≥0.02 and $\varphi\left(A_{k,1}, \ldots, A_{k,j}\right)$≥0.02.

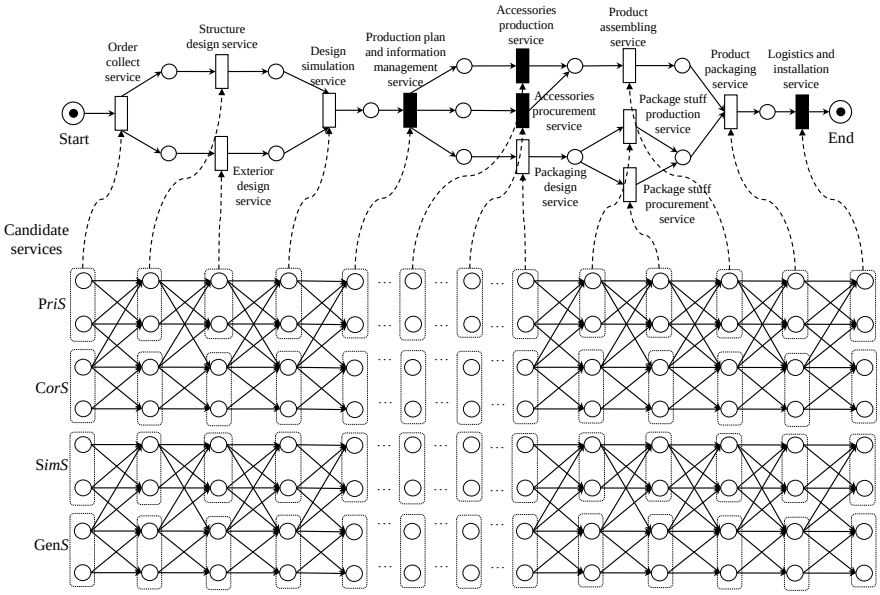
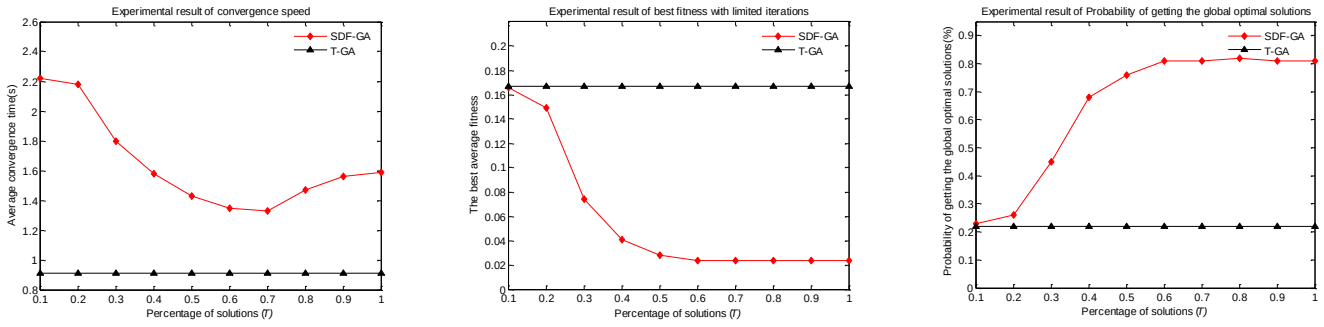


Fig. 4 Service process with subtasks for a service request for household electrical appliances

6.2 Case 1: Effect of SDFs on the proposed algorithm

This case study is designed to analyse the influence of different levels of richness characterizing the feature sets on the effectiveness of SDF-GA for solving SCOS problems and the working mechanism of different SDFs. Accordingly, two experiments are implemented. The first experiment uses different thresholds T to generate feature sets with different levels of richness and runs SDF-GA and T-GA to analyse their performance. The other experiment applies different algorithms derived by SDF-GA for analysing the working mechanism of the SDFs.

For the first experiment, T-GA is initially performed 200 times with 100 iterations to generate historical solutions. Different percentage values of historical solutions are used as a threshold T to divide the generated solution into positive and negative solutions for achieving feature sets with different levels of richness. For a certain percentage p, SDF-GA is performed 50 times with 100 iterations to comprehensively assess its convergence speed, capability for finding the best fitness with limited iterations and probability of obtaining the global optimal solution. For all individuals of SDF-GA and T-GA, their fitness values are normalized by Eq. 4, which assigns the best fitness value (the minimum cost) to 0 and the worst fitness value to 1. Moreover, solutions belonging to the set $F=\{f_i | f_{best}-f_i \leq \varepsilon=0.08\}$ (f_{best} is the best solution found by SDF-GA) are considered the global optimal solution. When the best solution found by the algorithms belongs to the set F, the number of times for finding the global optimal solution is increased. A comparison of the average results is shown in Fig. 5, in which the performance of T-GA is used as a baseline.



(a) Convergence speed comparison (b) Capability comparison (c) Probability comparison

Fig. 5 Effectiveness of SDF-GA with different levels of richness (p)

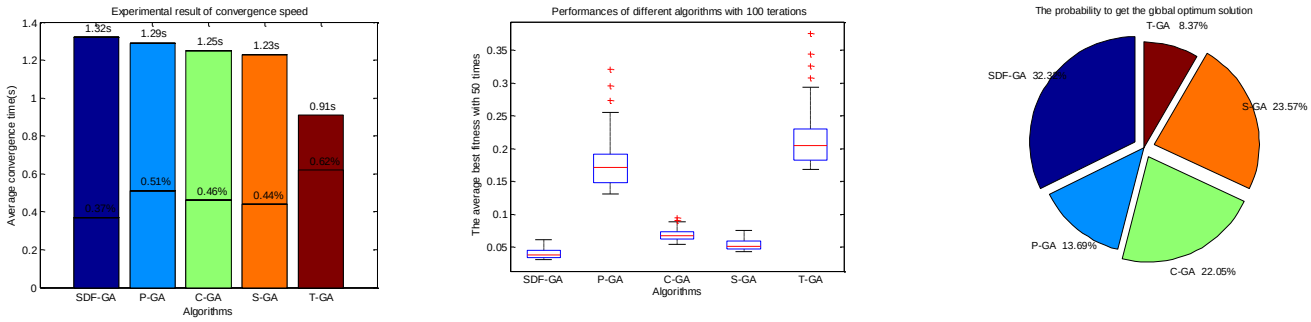
Fig. 5(a) shows the convergence speed results. The convergence speed of T-GA is the average value of the convergence time over 200 runs. With increasing percentage of positive solutions, the average convergence speed of SDF-GA initially increases but then decreases. When the levels of positive solutions are between 30 and 60 percent, the convergence speed increases dramatically. The reason for this phenomenon might be that in these cases, the feature sets can contain reusable solutions and more-superior candidate services. In other words, the probability that the feature sets will include the global optimal solution

increases. When the levels of the positive solutions are greater than 70 percent, the convergence speed decreases because the scale of the feature sets is larger, and more time is needed to search the subspaces.

Fig. 5(b) shows the capabilities for finding the best solutions with a limited number of iterations. The best average fitness increases with increasing amount of positive solutions. When the percentage of positive solutions is 60%, the best average fitness reaches a maximum and barely changes because when a feature set contains the necessary qualified candidate service for the global optimal solution, and SFDs-GA can always find this solution regardless of the scale of the feature sets (Fig. 5(a) indicates that a large scale might require more time). This characteristic is a benefit obtained from the fine-grained definitions of the SFDs. Fig. 5(c) shows a similar circumstance. When the percentage of positive solutions increases, the probability of obtaining the best optimal solution is maximized at 82% and changes only slightly.

Overall, with the appropriate feature sets, the performance of SDF-GA is found to be outstanding relative to T-GA. The experimental results show that the SDFs are not always valid until the richness of the SDFs reaches the baseline. These results are consistent with the discussion based on lemma 2.

Next, to independently assess the working mechanism of SDFs for the proposed algorithm, three algorithms are derived based on SDF-GA: improved GAs based on the prior (P-GA), correlation (C-GA), and similarity (S-GA). P-GA, C-GA and S-GA solely use the corresponding feature set. Based on the first experiment, 60% of the generated solutions are assigned as the threshold T and used to generate the feature sets. Each algorithm is implemented 50 times with 100 iterations for multiple analyses with respect to the convergence speed, the capability of finding the best solution with limited number of iterations and the probability of obtaining the global optimum solution. The experimental results are shown in Fig. 6.



(a) Convergence speed comparison (b) Capability comparison (c) Probability comparison

Fig. 6 Performance comparison of algorithms

Fig. 6(a) shows that the average number of iterations until convergence of SDF-GA is the lowest. However, the convergence time of SDF-GA is the highest (SDF-GA's average number of iterations until convergence is 37, and that of T-GA is 62) because each iteration of SDF-GA needs more time to generate new generations. The convergence performance of C-GA and S-GA is better than that of P-GA, which might indicate that the correlation and similarity features have a better effect on the convergence performance of SDF-GA. Fig. 6(b) shows the box graph with the average best fitness of each iteration from 0 to 100 – 50 times. The performance of SDF-GA in terms of finding the best solution is distinctly better, followed by S-GA and C-GA, which might further indicate that the correlation and similarity features strongly determine the performance of SDF-GA. The performance of P-GA is close to the performance of T-GA, which might be because the services in the prior set that have QoS correlations with other services have been moved to the correlation set; the remaining services in P*riS* can only be used to form local optimal solutions. P-GA has a greater chance to randomly search the space around the local optimal solution. The compared results illustrate that the three features can work together to make the SDF-GA perform well and that a single feature has only limited benefit. In addition, the outliers in Fig. 6(b) are the average best fitnesses of the initial iterations, which are much greater than the average values of the sequential iteration process. This result also shows the convergence states of the algorithms from another perspective; SDF-GA and S-GA can rapidly reach the converged state. S-GA has a better ability to find the global optimal solution than does C-GA or P-GA. Figure 7 (c) shows the probability of the algorithms to obtain the global optimal solution. This result further confirms the superiority of SDF-GA and benefits of the correlation and similarity with respect to the performance of SDF-GA.

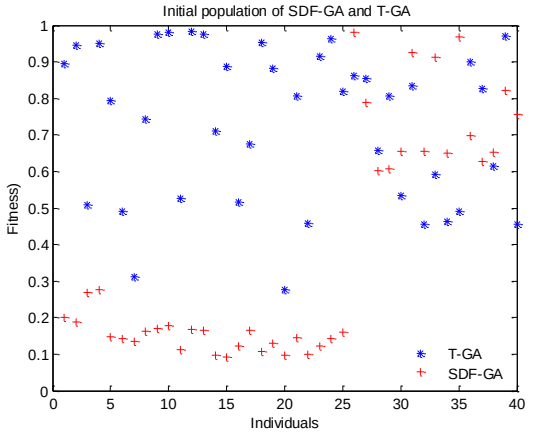
To summarize, the results have shown that similarity and correlation play larger roles in the performance of SDF-GA. Simultaneously, a single feature has limited effectiveness. Considering the dependencies among feature sets, these three features

should be used in combination to help solve SCOS problems. These results also provide evidence that proves **Corollaries 1 and** 2.

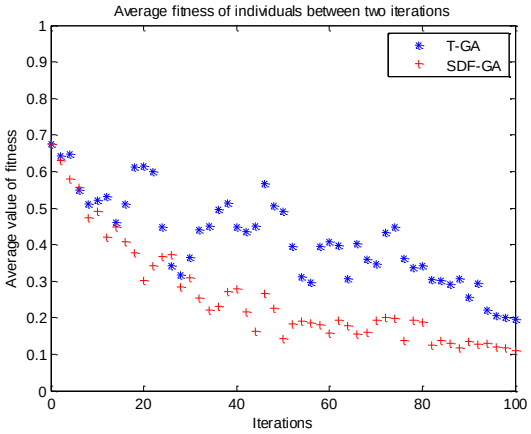
6.3 Case 2: Effectiveness of optimization strategies of proposed algorithm

This case study is designed to verify the effectiveness of optimization strategies for the proposed algorithm. Similarly, 60% is used as the threshold T to generate the feature sets. Algorithms 2, 3, 4 and 5 are performed with T-GA separately to analyse each strategy. Algorithms 3 and 4 are performed together with the normal initial population and normal fitness function of T-GA, and algorithm 5 is performed with a normal initial population and operators from T-GA. Comparisons of the experimental results are shown in Fig. 7. Fig. 7(a) shows the initial population generated by algorithm 2 and T-GA. Most individuals generated by algorithm 2 are initialized around the best solutions found by T-GA, which confirms the effectiveness of algorithm 2 and lemma 2. Simultaneously, individuals generated from C*orS* and S*imS* are better than individuals generated from P*riS*, which further indicates the main roles of correlation and similarity. Fig. 7(b) shows the average fitness of two iterations of T-GA and T-GA with algorithms 3 and 4. This result confirms lemma 3, that is, with the SDF-oriented operators, SDF-GA can quickly find the best solution. When SDF-GA runs for approximately 20 iterations, new generations of SDF-GA can reuse local optimal solutions generated by T-GA to search further for the global optimal solution. This point also suggests that SDF-GA's performance depends heavily upon the quality of the feature sets; thus, if the quality of the feature sets is not very high, then the sets can converge rapidly to a poor solution around the local optimal solution. This circumstance is consistent with the discussion of lemma 2. Fig. 7(c) contrastively shows the individual fitness of the final iterations of T-GA and SDF-GA, which have different fitness functions. As seen from the figure, the population of SDF-GA has a greater diversity than does T-GA, which implies the beneficial effect of fitness-sharing niching technology.

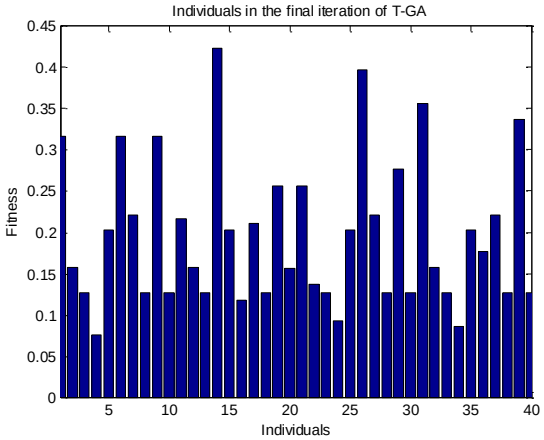
In summary, this case study verifies the effectiveness of optimization strategies. Combining with the results of the two case studies above, we can conclude that, compared with T-GA, SDF-GA achieves a higher performance in solving SCOS problems.

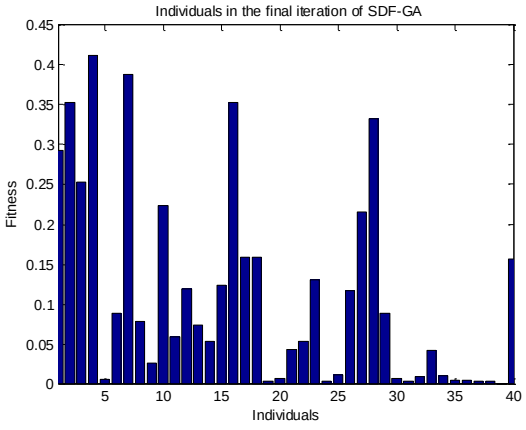


(a) Effectiveness of initial population



(b) Effectiveness of operators





(c) Effectiveness of fitness function

Fig. 7 Effectiveness of optimization strategies of SDF-GA

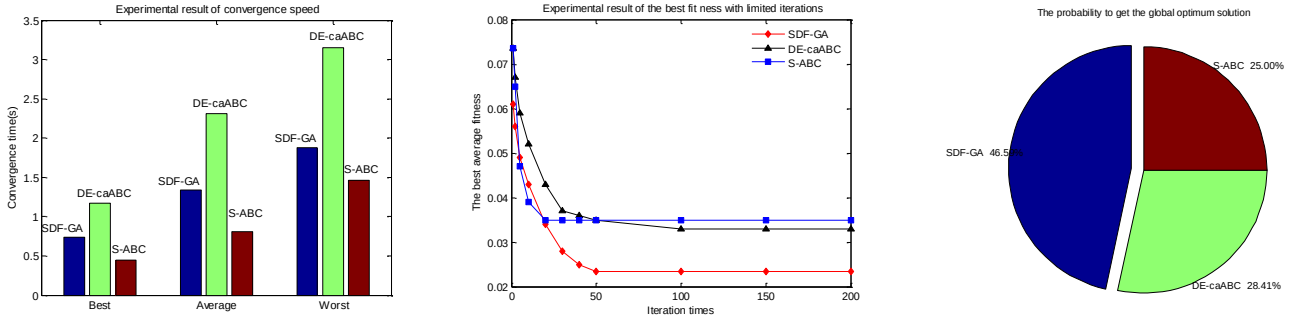
6.4 Case 3: Search ability of proposed algorithm compared with other SDF-oriented algorithms

This case study is designed to assess the performance of SDF-GA compared with DE-caABC and S-ABC, which are two SDF-oriented algorithms and have been shown to be superior to traditional algorithms such as GA, ABC, and AC. The settings of DE-caABC and S-ABC are the original settings in the corresponding studies. These two algorithms were first performed 50 times with 200 iterations on the dataset. The comparison results with SDF-GA are shown in Fig. 8. Moreover, to fully assess the performance of the algorithms, five datasets were synthesized based on a pre-existing dataset. The synthesized datasets are shown in Table 6. The first dataset is the pre-existing dataset. For each synthesized set, the GA also is used to generate a solution to obtain feature sets with the 60% threshold. Then, these three algorithms are performed 50 times with 200 iterations each. The experimental results are shown in Fig. 9.

Fig. 8(a) shows the convergence speed of the algorithms; the convergence speed of SDF-GA is better than that of DE-caABC and worse than that of S-ABC. Fig. 8(b) shows the best average fitness; the performance of SDF-GA is the highest. S-ABC can rapidly converge, possibly because abundant prior schemas are used in S-ABC and because there is little focus on superior similarity and QoS correlation. DE-caABC suffers from the same drawbacks, but the DE stage can help DE-caABC find better solutions based on the generated local optimal solutions. Fig. 8(c) shows the comparative probabilities of obtaining the global optimal solution. As seen from the figure, the performance of SDF-GA is the best, and DE-caABC is better than S-ABC.

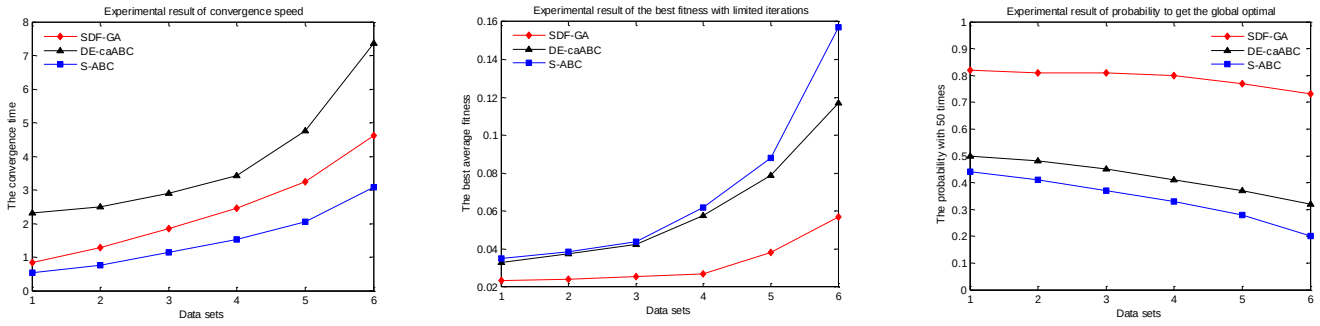
Fig. 9 shows that with increasing number of subtasks and candidate services, the performance of DE-caABC and S-ABC with respect to the three metrics decreases dramatically. In contrast, the performance of SDF-GA presents a slow decline. SDF-GA performs well on all synthesized datasets, which indicates that the proposed algorithm has the better searching capability both in local and global search.

In summary, the above experimental results prove the superiority of SDF-GA. It has better local and global searching abilities, whereas DE-caABC and S-ABC are similar to the P-GA, which focusses on local searching and has a better local search ability. Consequently, the proposed algorithm can more effectively solve SCOS problems.



(a) Convergence speed comparison (b) Capability comparison (c) Probability comparison

Fig. 8 Algorithm performance comparison



(a) Convergence speed comparison (b) Capability comparison (c) Probability comparison

Fig. 9 Algorithm performance comparison with different datasets

Table 5 Details of the synthetic datasets

No.	1	2	3	4	5	6
Subtasks	72	100	120	150	180	200
Candidate service	58	100	150	200	250	300

6.5 Discussion

With the above experiments, SDF-GA is shown to achieve good performance, which is in agreement with the theoretical analysis. Compared with T-GA and other SDF-oriented algorithms, given a certain amount of prior knowledge, it achieves superior performance in terms of convergence speed (convergence iterations), a capability for finding the best solution and a probability of finding the global optimal solution. Similarity and correlation play important roles for SDF-GA in finding the global optimal solution and might be the key reason that SDF-GA can perform better than other SDF-oriented algorithms. However, with the dependencies among feature sets, these features should be used together to obtain the best performance. The scale of the feature sets can influence the convergence speeds but has only a slight effect on the capability of finding the best solution and probability of obtaining globally optimal solutions. Figs. 5 (b) and (c) show that when the percentage is greater than 30%, the probability of feature sets containing services of the global optimal solution increases and SDF-GA gradually begins to perform well; when the feature sets cover sufficient services for the global optimal solution, SDF-GA achieves its top performance. The better local and global searching abilities of SDF-GA rely heavily on the quality of the feature sets, which means that the threshold T of the utility function should be set with an appropriate value for acquiring sufficient prior knowledge. Moreover, thresholds for generating feature sets should be adjusted with the threshold T.

The superiority of SDF-GA makes it suitable for the SCOS in CMfg, particularly for a service composition with many subtasks and candidate services. For such SCOS problems, SDF-GA can obtain the optimal solution efficiently and effectively with a certain amount of prior knowledge. For SCOS problems without historical data, T-GA and other intelligent optimization algorithms can first be used to generate the prior knowledge for setting up the SDF-GA to obtain a more optimal solution. With the help of SDF-GA and the easily attained SDFs from historical practices, manufacturers who are willing to utilize CMfg for implementing mass customization or the new industrial revolution based on flexible service composition can arrive at their goals safely and efficiently, with high-quality compositional services and products.

Conclusion and future work

SCOS problems in CMfg are becoming more complicated, with ever-increasing candidate services and the innovation of an intelligent manufacturing mode. In this paper, a service domain feature-oriented genetic algorithm is proposed in response to the deficiency of existing methods in addressing this problem. More specifically, some fine-grained definitions of Service Domain Features (SDFs) are presented, and an algorithm for dividing the services into four subspaces with defined features is proposed. Then, with the divided service space, optimization strategies of the initial population, operators and fitness function based on SDFs are presented. The theoretical analysis of the property of proposed algorithm and its superiority is further provided based on the Bayes theorem. Its effectiveness and efficiency are verified through three case studies of SCOS problems in CMfg. The proposed algorithm can be used not only for manufacturers who are attempting to employ service composition in CMfg to address the pressure of lower costs and to improve the core competitive ability in the current “globalization of manufacturing” environment but also for providing some theoretical guidance and insights to CMfg designers.

The main contributions of this paper are summarized as follows:

(1) With the purpose of using priori knowledge to solve SCOS problems, we defined some fine-grained definitions of SDFs that can be used to reasonably divide the service space of SCOS problems.

(2) To quickly and thoroughly search the divided service space in obtaining optimal solutions, we proposed a service domain feature-oriented genetic algorithm with SDF-based optimization strategies. The performance comparisons indicated that the proposed algorithm is better than other SDF-based algorithms for solving SCOS problems, especially for solving SCOS problems with a large-scale search space.

(3) We studied the proposed algorithm. Using the theory of the Bayes theorem, its superiority in terms of the performance with respect to convergence speed and optimization result is proved. The experimental results showed that the results were in accordance with the theoretical analysis.

In the future, our work will further refine the definitions of SDFs and strategies by studying the division of subspaces for specific SCOS problems. We will apply the proposed algorithm to solve SCOS problems in several practical manufacturing

domains (e.g., automobile manufacturing, electric manufacturing, and zipper manufacturing) to further explore its practical value.

Acknowledgment

This work has been supported in part by the research projects of the National Natural Science Foundation of China (NSFC) (No. 71571056) and the Scientific Research Funds of Huaqiao University (16BS304).

Appendix

1. Proof of Lemma 2. A set of values of $P_{T-GA}\left(B/A_h^{(0)}\right)$ can be derived based on the formula in **Property 1** for calculating $P_{T-GA}\left(B/A_h^{(0)}\right)$ and sorted in ascending order $0 \le P_{T-GA}\left(B/A_1^{(0)}\right) \le P_{T-GA}\left(B/A_2^{(0)}\right) \le \cdots \le P_{T-GA}\left(B/A_h^{(0)}\right)$, and then, we randomly split the set of values as $0 \le P_{T-GA}\left(B/A_1^{(0)}\right) \le P_{T-GA}\left(B/A_2^{(0)}\right) \le \cdots \le P_{T-GA}\left(B/A_{h-u}^{(0)}\right) \le P_{T-GA}\left(B/A_1^{(0)}\right) \le P_{T-GA}\left(B/A_2^{(0)}\right) \le \cdots \le P_{T-GA}\left(B/A_u^{(0)}\right)$. We ensure that the optimal subspace is one of subspaces among *u* subspaces with larger values of $P_{T-GA}\left(B/A_u^{(0)}\right)$ and use all solutions from *u* subspaces to initialize the population of SDF-GA. Based on Property 1, we have

$$P_{SDF-GA}\left(A^{(0)}\right) = \sum_{j=1}^{u} P_{SDF-GA}\left(A_j^{(0)}\right) = \frac{\sum_{j=1}^{u} P_{T-GA}\left(B/A_j^{(0)}\right) \times P_{T-GA}\left(A_j^{(0)}\right)}{\sum_{i=1}^{h-u} P_{T-GA}\left(B/A_i^{(0)}\right) \times P_{T-GA}\left(A_i^{(0)}\right) + \sum_{j=1}^{u} P_{T-GA}\left(B/A_j^{(0)}\right) \times P_{T-GA}\left(A_j^{(0)}\right)}$$

$$\frac{\sum_{j=1}^{u} \frac{P_{T-GA}\left(B/A_j^{(0)}\right)}{minP_{T-GA}\left(B/A_j^{(0)}\right)} \times P_{T-GA}\left(A_j^{(0)}\right)}{\sum_{i=1}^{h-u} \frac{P_{T-GA}\left(B/A_i^{(0)}\right)}{minP_{T-GA}\left(B/A_j^{(0)}\right)} \times P_{T-GA}\left(A_i^{(0)}\right) + \sum_{j=1}^{u} \frac{P_{T-GA}\left(B/A_j^{(0)}\right)}{minP_{T-GA}\left(B/A_j^{(0)}\right)} \times P_{T-GA}\left(A_j^{(0)}\right)}$$

Because of this, the value of $P_{T-GA}\left(B/A_i^{(0)}\right)$ is smaller than the value of $minP_{T-GA}\left(B/A_j^{(0)}\right)$, and thus, we can have

$$P_{SDF-GA}\left(A^{(0)}\right) \ge \frac{\sum_{j=1}^{u} \frac{P_{T-GA}\left(B/A_j^{(0)}\right)}{minP_{T-GA}\left(B/A_j^{(0)}\right)} \times P_{T-GA}\left(A_j^{(0)}\right)}{\sum_{i=1}^{h-u} P_{T-GA}\left(A_i^{(0)}\right) + \sum_{j=1}^{u} \frac{P_{GA}\left(B/A_j^{(0)}\right)}{minP_{GA}\left(B/A_j^{(0)}\right)} \times P_{T-GA}\left(A_j^{(0)}\right)}$$

$$\frac{\sum_{j=1}^{u} \frac{P_{T-GA}\left(B/A_j^{(0)}\right)}{minP_{T-GA}\left(B/A_j^{(0)}\right)} \times P_{T-GA}\left(A_j^{(0)}\right)}{1 - \sum_{j=1}^{u} P_{T-GA}\left(A_j^{(0)}\right) + \sum_{j=1}^{u} \frac{P_{T-GA}\left(B/A_j^{(0)}\right)}{minP_{T-GA}\left(B/A_j^{(0)}\right)} \times P_{T-GA}\left(A_j^{(0)}\right)}$$

$$\frac{\sum_{j=1}^{u} P_{T-GA}\left(A_j^{(0)}\right) + \sum_{j=1}^{u}\left(\frac{P_{T-GA}\left(B/A_j^{(0)}\right)}{minP_{T-GA}\left(B/A_j^{(0)}\right)} - 1\right) \times P_{T-GA}\left(A_j^{(0)}\right)}{1 + \sum_{j=1}^{u}\left(\frac{P_{T-GA}\left(B/A_j^{(0)}\right)}{minP_{T-GA}\left(B/A_j^{(0)}\right)} - 1\right) \times P_{T-GA}\left(A_j^{(0)}\right)}$$

Because of $\sum_{i=1}^{h-u} P_{T-GA}\left(A_i^{(t)}\right) + \sum_{j=1}^{u} P_{T-GA}\left(A_j^{(t)}\right) = 1$, we have

$$P_{SDF-GA}\left(A^{(0)}\right) = \frac{1 + \sum_{j=1}^{u}\left(\frac{P_{T-GA}\left(B/A_j^{(0)}\right)}{minP_{T-GA}\left(B/A_j^{(0)}\right)} - 1\right) \times P_{GA}\left(A_j^{(0)}\right) - \sum_{i=1}^{h-u} P_{T-GA}\left(A_i^{(0)}\right)}{1 + \sum_{j=1}^{u}\left(\frac{P_{T-GA}\left(B/A_j^{(0)}\right)}{minP_{T-GA}\left(B/A_j^{(0)}\right)} - 1\right) \times P_{T-GA}\left(A_j^{(0)}\right)}$$

$$1 - \frac{\sum_{i=1}^{h-u} P_{T-GA}\left(A_i^{(0)}\right)}{1 + \sum_{j=1}^{u}\left(\frac{P_{T-GA}\left(B/A_j^{(0)}\right)}{minP_{T-GA}\left(B/A_j^{(0)}\right)} - 1\right) \times P_{T-GA}\left(A_j^{(0)}\right)}$$

Because of $\sum_{j=1}^{u}\left(\frac{P_{T-GA}\left(B/A_j^{(0)}\right)}{minP_{T-GA}\left(B/A_j^{(0)}\right)} - 1\right) \times P_{T-GA}\left(A_j^{(0)}\right) \geq 0$, we have

$$P_{SDF-GA}\left(A^{(0)}\right) \geq 1 - \sum_{i=1}^{h-u} P_{-A}\left(A_i^{(0)}\right) = \sum_{j=1}^{u} P_{T-GA}\left(A_j^{(0)}\right) = P_{T-GA}\left(A^{(0)}\right)$$

2. Proof of Lemma 3. Based on Property 1, we have

$$P_{T-GA}\left(A^{(1)}\right) = \frac{\sum_{j=1}^{u} P\left(B/A_j^{(0)}\right) P\left(A_j^{(0)}\right)}{\sum_{i=1}^{h-u} P\left(B/A_i^{(0)}\right) P\left(A_i^{(0)}\right) + \sum_{j=1}^{u} P\left(B/A_j^{(0)}\right) P\left(A_j^{(0)}\right)}$$

$$\frac{1}{\frac{\sum_{i=1}^{h-u} P\left(B/A_i^{(0)}\right) P\left(A_i^{(0)}\right)}{\sum_{j=1}^{u} P\left(B/A_j^{(0)}\right) P\left(A_j^{(0)}\right)} + 1}$$

Because the operators of SDF-GA use feature sets generated from $A_{GA}^{(1)}$, we have

$$P_{SDF-GA}\left(A^{(1)}\right) = \frac{\sum_{j=1}^{u} P_{T-GA}\left(B/A_j^{(1)}\right) P\left(A_j^{(0)}\right)}{\sum_{i=1}^{h-u} P_{T-GA}\left(B/A_i^{(1)}\right) P\left(A_i^{(1)}\right) + \sum_{j=1}^{u} P_{T-GA}\left(B/A_j^{(1)}\right) P_{GA}\left(A_j^{(1)}\right)}$$

$$\frac{1}{\frac{\sum_{i=1}^{h-u} P_{T-GA}\left(B/A_i^{(1)}\right) P\left(A_i^{(1)}\right)}{\sum_{j=1}^{u} P_{T-GA}\left(B/A_j^{(1)}\right) P\left(A_j^{(1)}\right)} + 1}$$

Based on **Lemma 2**, we can obtain $\sum_{j=1}^{u} P_{GA}\left(B/A_j^{(1)}\right) P\left(A_j^{(1)}\right) \geq \sum_{j=1}^{u} P_{GA}\left(B/A_j^{(0)}\right) P\left(A_j^{(0)}\right)$, and thus, we have $P_{SDF-GA}\left(A^{(1)}\right) \geq P_{GA}\left(A^{(1)}\right)$.

References

Bravo, M. (2014). Similarity measures for web service composition models. *International Journal on Web Service Computing*, 495-505.

Chen, F., Dou, R., Li, M., & Wu, H. (2016). A flexible QoS-aware Web service composition method by multi-objective optimization in cloud

manufacturing. *Computers & Industrial Engineering*, 99, 423-431.

Chen, R., Guo, J., & Bao, F. (2016). Trust management for SOA-based IoT and its application to service composition. *IEEE Transactions on Services Computing*, *9*(3), 482-495.

Fatahi Valilai, O., & Houshmand, M. (2014). A platform for optimisation in distributed manufacturing enterprises based on cloud manufacturing paradigm. *International Journal of Computer Integrated Manufacturing*, *27*(11), 1031-1054.

Feng, Y., & Huang, B. (2018). Cloud manufacturing service qos prediction based on neighbourhood enhanced matrix factorization. *Journal of Intelligent Manufacturing* (1), 1-12.

Hua, G., Zhang, L., Liu, Y., Tao, F., Shu, M., & Mu, S. (2014). A discovery method of service-correlation for service composition in virtual enterprise. *European Journal of Industrial Engineering*, *8*(5), 579-618.

Huang, B., Li, C., & Tao, F. (2014). A chaos control optimal algorithm for QoS-based service composition selection in cloud manufacturing system. *Enterprise Information Systems*, 8(4), 445-463.

Huang, J., Li, S., Duan, Q., Yu, R., & Yu, S. (2016, December). QoS Correlation-Aware Service Composition for Unified Network-Cloud Service Provisioning. In *Global Communications Conference (GLOBECOM), 2016 IEEE* (pp. 1-6). IEEE.

Jiang, Y. Z., Hao, Z. F., Zhang, Y. S., Huang, H., Wang, Y. L., & He, H. J. (2014). Bayesian forecasting evolutionary algorithm. *Chinese Journal of Computers*.

Jin, H., Yao, X., & Chen, Y. (2017). Correlation-aware QoS modeling and manufacturing cloud service composition. *Journal of Intelligent Manufacturing*, 28(8), 1947-1960.

Kai, C., Guohu, C., & Hua, J. (2014). Guided self-adaptive evolutionary genetic algorithm. *Journal of Electronics & Information Technology*, 36(8), 1884-1890.

Karim, R., Ding, C., & Miri, A. (2015, June). End-to-end QoS prediction of vertical service composition in the cloud. In *Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on* (pp. 229-236). IEEE.

Kubler, S., Holmström, J., Främling, K., & Turkama, P. (2016). *Technological Theory of Cloud Manufacturing*. *Service Orientation in Holonic and Multi-Agent Manufacturing*. Springer International Publishing.

Lemos, A. L., Daniel, F., & Benatallah, B. (2016). Web service composition: a survey of techniques and tools. *ACM Computing Surveys (CSUR)*, 48(3), 33.

Li BH, Zhang L,Wang SL, Tao F, Cao JW, JiangXD, SongX, Chai XD (2010) Cloud manufacturing: a new service-oriented networked manufacturing model. *Comput Integr Manuf Syst* 16(1):1–16

Liu, J., Hao, S., Zhang, X., Wang, C., Sun, J., Yu, H., & Li, Z. (2016, June). Research on Web Service Dynamic Composition Based on Execution Dependency Relationship. In *Services (SERVICES), 2016 IEEE World Congress on* (pp. 113-117). IEEE.

Liu, Z., & Xu, X. (2014, June). S-ABC-A Service-oriented artificial bee colony algorithm for global optimal services selection in concurrent requests environment. In *Web Services (ICWS), 2014 IEEE International Conference on* (pp. 503-509). IEEE.

Lu, Y., & Xu, X. (2017). A semantic web-based framework for service composition in a cloud manufacturing environment. *Journal of Manufacturing Systems*, 42, 69-81.

Morgan, J., & O'Donnell, G. E. (2017). Enabling a ubiquitous and cloud manufacturing foundation with field-level service-oriented architecture. *International Journal of Computer Integrated Manufacturing*, *30*(4-5), 442-458.

Pisching, M. A., Junqueira, F., Filho, D. J. S., & Miyagi, P. E. (2015). *Service Composition in the Cloud-Based Manufacturing Focused on the Industry 4.0. Technological Innovation for Cloud-Based Engineering Systems*. Springer International Publishing.

Seghir, F., & Khababa, A. (2016). A hybrid approach using genetic and fruit fly optimization algorithms for QoS-aware cloud service composition. *Journal of Intelligent Manufacturing*, 1-20.

Tao, F., Cheng, Y., Da Xu, L., Zhang, L., & Li, B. H. (2014). CCIoT-CMfg: cloud computing and internet of things-based cloud manufacturing service system. *IEEE Transactions on Industrial Informatics*, *10*(2), 1435-1442.

Tao, F., Cheng, Y., Zhang, L., & Nee, A. Y. C. (2017). Advanced manufacturing systems: socialization characteristics and trends. *Journal of Intelligent Manufacturing, 28*(5), 1079-1094.

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 Transactions on Industrial Informatics*, 9(4), 2023-2033.

Tao, F., Zhang, L., Liu, Y., Cheng, Y., Wang, L., & Xu, X. (2015). Manufacturing service management in cloud manufacturing: overview and future research directions. *Journal of Manufacturing Science and Engineering*, *137*(4), 040912.

Tao, F., Zhao, D., Yefa, H., & Zhou, Z. (2010). Correlation-aware resource service composition and optimal-selection in manufacturing grid. *European Journal of Operational Research*, *201*(1), 129-143.

Tao, F., Zuo, Y., Da Xu, L., & Zhang, L. (2014). IoT-based intelligent perception and access of manufacturing resource toward cloud manufacturing. *IEEE Transactions on Industrial Informatics*, *10*(2), 1547-1557.

Van Nguyen, S., Vo, H. D., & Hung, P. N. (2015, December). A Correlation-aware Negotiation Approach for Service Composition. In *Proceedings of the Sixth International Symposium on Information and Communication Technology* (pp. 210-216). ACM.

Wang, J., Zhang, L., Duan, L., & Gao, R. X. (2017). A new paradigm of cloud-based predictive maintenance for intelligent manufacturing. *Journal of Intelligent Manufacturing*, 28(5), 1125-1137.

Wu, Q., Zhu, Q., & Zhou, M. (2014). A correlation-driven optimal service selection approach for virtual enterprise establishment. *Journal of Intelligent Manufacturing*, *25*(6), 1441-1453.

Xiang, F., Jiang, G., Xu, L., & Wang, N. (2016). The case-library method for service composition and optimal selection of big manufacturing data in cloud manufacturing system. The International Journal of Advanced Manufacturing Technology, 84(1-4), 59-70.

Xu, X. (2012). From cloud computing to cloud manufacturing. *Robotics and computer-integrated manufacturing*, *28*(1), 75-86.

Xu, X., Liu, Z., Wang, Z., Sheng, Q. Z., Yu, J., & Wang, X. (2017). S-ABC: A paradigm of service domain-oriented artificial bee colony algorithms for service selection and composition. *Future Generation Computer Systems*, 68, 304-319.

Xue, X., Liu, Z. Z., & Wang, S. F. (2016). Manufacturing service composition for the mass customised production. *International Journal of Computer Integrated Manufacturing*, *29*(2), 119-135.

Ye, Z., Mistry, S., Bouguettaya, A., & Dong, H. (2016). Long-term qos-aware cloud service composition using multivariate time series analysis. *IEEE Transactions on Services Computing, 9*(3), 382-393.

Zhang, M. W., Wei, W. J., Zhang, B., Zhang, X. Z., & Zhu, Z. L. (2008). Research on service selection approach based on composite service execution information. *Chinese Journal of Computers*, *31*(8), 1398-1411.

Zhang, S., Xu, Y., Zhang, W., & Yu, D. (2017). A new fuzzy qos-aware manufacture service composition method using extended flower pollination algorithm. *Journal of Intelligent Manufacturing* (4), 1-15.

Zhang, Y., Zhang, G., Liu, Y., & Hu, D. (2017). Research on services encapsulation and virtualization access model of machine for cloud manufacturing. *Journal of Intelligent Manufacturing*, *28*(5), 1109-1123.

Zheng, H., Feng, Y., & Tan, J. (2016). A fuzzy QoS-aware resource service selection considering design preference in cloud manufacturing system. *International Journal of Advanced Manufacturing Technology*, 84 (1-4), 371-379.

Zhou, J., & Yao, X. (2017). DE-caABC: differential evolution enhanced context-aware artificial bee colony algorithm for service composition and optimal selection in cloud manufacturing. *The International Journal of Advanced Manufacturing Technology*, 90(1-4), 1085-1103.