IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received January 5, 2020, accepted January 20, 2020, date of publication January 24, 2020, date of current version January 31, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2969234

Composition of Resource-Service Chain Based on Evolutionary Algorithm in Distributed Cloud Manufacturing Systems

HAIBO LI[1,2], SHAOYUAN WENG[1], JUNCHENG TONG[1], TING HE[1], WENYUN CHEN[1], MENGMENG SUN[1], AND YINGTONG SHEN[1]

[1]College of Computer Science and Technology, Huaqiao University, Xiamen 361021, China
[2]Xiamen Engineering Research Center of Enterprise Interoperability and Business Intelligence, Xiamen 361021, China

Corresponding author: Haibo Li (lihaibo@hqu.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB1402500, in part by the Fujian Province Science and Technology Plan under Grant 2019H0017, and in part by the Quanzhou Science and Technology Plan under Grant 2018C110R and Grant 2018Z008.

ABSTRACT In distributed cloud manufacturing (CMfg) systems, multi-resource service can complete more complex manufacturing tasks than single resource service. Especially in business process, all the resource services are invoked in a certain sequence, which is called the Resource-Service Chain (RSC). The RSC, as a sequential composition of resource services, expresses the scheduling and the flow of servicing to a distributed business process. A perfect composition can improve utilization ratio and efficient matching availability of resource services greatly. However, most of the existing methods for resource service composition paid no attention to the temporal relationship between resource services. Moreover, the methods strongly depend on relevant element to be considered. Inspired by biological evolution, a Resource-Service Chain Composition Evolutionary (RSCCE) algorithm is proposed. Specifically, RSCCE tries to find multiple optimal solutions, namely all RSCs in a workflow with given constraints. To begin, initial sets of composite resource service are resolved by calculating the degree of dependency between resource services, so as to obtain initial RSCs by workflow. Then, RSCCE algorithm applies genetic algorithm to search for the extended of each initial RSC, a longer chain composing of it, to improve the reuse of RSC. Under this approach, gene and chromosome represent resource service and the entire RSC respectively. If the propagated chromosomes violate the sequence of resource service, as constraint in RSCCE algorithm, they will be repaired to obtain a valid solution. Finally, we take a multi-enterprise collaborative business process as an example to simulate our approach. Experimental results confirm the effectiveness of the approach.

INDEX TERMS Distributed cloud manufacturing, resource-service chain, composition, evolutionary algorithm, business process.

I. INTRODUCTION

Cloud Manufacturing (CMfg) is a new distributed network manufacturing supplying all kinds of manufacturing services on demand. All dispersed and diverse manufacturing resources from different enterprises are encapsulated into cloud services, organized and integrated into CMfg service platform under the support of distributed computing and cloud computing and Internet of Things technologies. Accordingly, the resource services can be managed and operated in an intelligent and unified way, can be provided to users in a united and centralized way to enable full sharing and circulation of manufacturing resources and capabilities [1]–[3].

In order to complete more complex manufacturing tasks, existing single resource services should be organized as composite resource service (CRS), being invoked as a whole, to provide more efficient and better value-added resource services [4] and integrate enterprises [5], [6]. For example, in a collaborative design and manufacturing processes of electrical apparatus, the task *hardware design* in the business process needs a CRS composed of three single

The associate editor coordinating the review of this manuscript and approving it for publication was Songwen Pei.

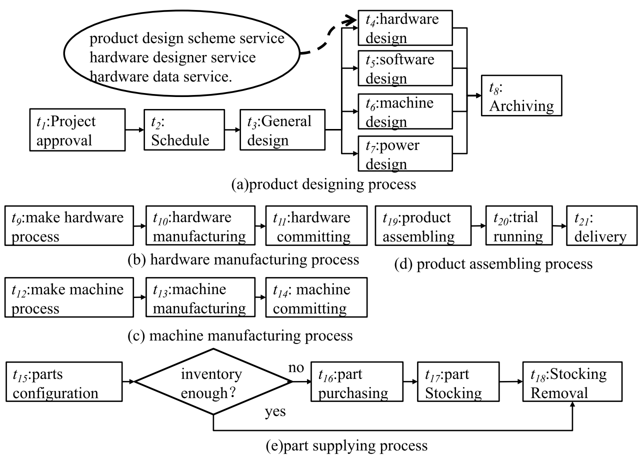


FIGURE 1. Collaborative design and manufacturing processes of electrical apparatus.

resource services, as shown in Fig. 1. This is called resource service composition. However in CMfg, massive and various manufacturing resources are published by enterprises of different domains, including hardware, software, human resources, data and information, etc. CRS should satisfy the requirements of different industrial features [4]. In view of these facts, business process, as functional representation to responsibility scope of domains, should be taken into consideration in resource service composition at least.

The researches on resources composition for distributed computing systems primarily concentrate on concept and method, despite of a relatively long history of manufacturing modes, including distributed manufacturing, dispersed network manufacturing, virtual enterprises and cloud manufacturing/manufacturing-as-a-service [7], [8]. Of all the existing methods for resource service composition, agent-based methods have been used for resource composition, in which agent is regarded as integration of resource and behavior to provide CRS [9], [10]. Semantics-based composition are very different approach to the above, in which applications of ontology have been found in describing resources services [11]–[13], in describing semantic Web service compositions [14], and in modeling manufacturing resource holons [3]. In [13], web services are invoked to compose the domain web services. In [14], Ontology Web Language for Services (OWL-S) described the properties and capabilities of services in an unambiguous computer-interpretable form. The approaches that are based on QoS (Quality of Service) are proposed to solve resource composition [7], [15]–[17]. Their main purpose is for optimal allocation and optimal selection of various manufacturing resources and capabilities, using the business or non-functionality indicators of QoS. In addition to the approaches, a Petri Net based model [14], [18], a graph-based model [19], a formal privacy model [20] and a Hidden Markov Model [21] are also used for resource composition.

Other than the above, it should be noted that some of the existing researches have already taken composite sequence into consideration, that is to say, constructing CRS which composed of resources in invoked sequence. In [4], [15] and [17], the optimal resource services are selected from candidate resource services according to QoS indicators, and then composed following certain sequence. In [22], a recommendation of service chains was proposed. Another kind of composition is for the purpose of detecting a business anomaly [23] and correlation coefficient matching [24].

However, as described above, current works on resource composition are mainly based on agents, semantics, QoS and model. The resource service composition they considered more was a kind of spatial integration, but more attention was not paid to sequential composition. Generally, CMfg service platform uses workflow technology to achieve cloud manufacturing resources rapid sharing and efficient coordination [25]. However, workflow model do not capture when a given resource allocated to a task will be available at runtime. Consequently all resources have to be kept available until the end of the business process. This leads to an inefficient use of resources. Therefore, resource composition should be the perspective of the scheduling and the flow of services to a business process. As resources are more variable and dynamic in distributed manufacturing environment, if some of them become unavailable after a business process has started, users still have chance to reselect or reschedule the succeeding resource services in RSCs before the related task starts.

If resource services are invoked in a certain sequence, they form a chain of resource services, called the Resource-Service Chain (RSC). We call this problem resource service chains composition (RSCC). RSCC is to find the optimal RSCs from resource services invoked by business process, depending on the inter-dependencies between resource services, not on others factors beyond workflow model, such as QoS indicator. Therefore, RSCC, as a key technology for resource selection and resource service recommendation, should be a more general approach, and has not been adequately addressed.

With the consideration of the above problems and situations, inspired by biological evolution, a new method, namely resource-service chain composition evolutionary (RSCCE) algorithm is proposed to resolve the composition of RSC in CMfg. The problem of identifying RSC, which is the most efficiency in execution, is an optimizing searching problem. We use genetic algorithm (GA) to resolve the potential and available RSCs, which satisfy the sequence constraints of workflow. In our evolutionary algorithm, all possible RSCs from workflow model are encoded in chromosome. The composition of RSC strategy is divided into two stages, building initial RSC and optimization of extended RSC. At the first stage, we resolve CRSs with high dependencies according to task-related dependencies between resource services in a workflow by a statistical algorithm, and obtain the initial RSCs by workflow model. At the second stage, to improve the reuse of RSC, we resolve the extended RSC by our evolutionary algorithm RSCCE, which can evolve chromosomes with different lengths.

The rest of the paper is organized as follows. In Section II, we describe the characteristics of the RSCC that we are addressing. The formulation of RSCC problem is also described for resolving the RSCs. The composition of RSC strategy and an algorithm supporting it are also presented. Section III presents the RSCCE in detail. The RSCCE has been evaluated using different data sets. The evaluation criteria, the design of the experiments and the results are presented in Section IV. Finally, in Section V, we present a summary and discuss what we intend to do as future work.

II. PROBLEM FORMULATION

A. FEATURE OF MANUFACTURING RESOURCE SERVICES

Cloud manufacturing aims to perform large-scale collaboration for complex manufacturing by sharing distributed manufacturing resources. Resources are encapsulated as cloud services and deployed to the cloud service platform, where manufacturing resources can be shared and accessed by heterogeneous applications. Workflow technology, as a standard solution for business process management, is widely used to integrate distributed tasks and resource services [1]. Therefore in CMfg system, RSC is the sequence in which resource services are used by workflow.

Before introducing the RSCC problem, an example is presented to illustrate the method, as shown in Fig. 1. The process *collaborative design and manufacturing processes of electrical apparatus*, as a classic example of inter-enterprise collaboration, is presented. The process has to be executed by invoking several resource services, including machine part delivery services, power supply services, various design services and human resource services, etc. Some tasks need more than one resource service. For example, the task *hardware design* needs product design scheme service, hardware designer service and hardware data service. These three resource services can be composed as a CRS, which is only a composition without considering sequences between them, to complete any task *hardware design* in another similar business processes. Along with the execution of a workflow instance, sequence between CRSs is formed resulting from temporal orders between resource services invoked by tasks. Also as it is possible that one resource service is invoked by different tasks in workflow, maybe there exist "shorter" sequences in a CRS. Therefor what we pay more attention is not only CRS, but also the sequences in CRS, even those in the entire workflow. In the example, the resource service *design scheme service* and *hardware data service* are invoked by the succeeding task *product assembling*. RSCC problem is to find all RSCs, each of which describes the temporal order of single resource services.

B. PROBLEM STATEMENT

Workflow can be considered as a technical context of business process. Workflow model is used to prescribe the execution precedence of tasks. However what we emphasize is the inter-enterprise collaboration in CMfg. By analyzing above,

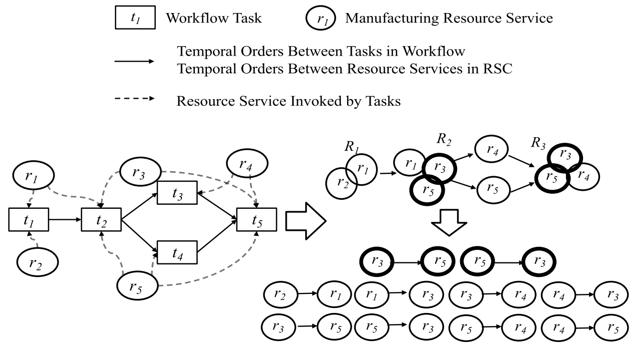


FIGURE 2. Resource service chain in workflow.

any workflow served to one of enterprises is a part of entire workflow, so that as a whole, all these workflows execute in sequence or parallel model. This is the design consideration for our method. Formally, workflow can be defined as follows.

Definition 1: workflow. A workflow *Wf*, is a 4-tuple <*id*, *Task*, $\prec$, *RS*>, reflecting the execution of a manufacturing process, where *id* is a unique identifier of the workflow, *Task* is a set of tasks, $\prec$ is a temporal order between tasks such that for any tasks t_p, $t_q \in Task$, $t_p \prec t_q$ indicates that t_p is executed before t_q, i.e t_p precedes t_q in a workflow instance. *RS* is a set of resource services and $RS = RS_1 \cup RS_2 \ldots \cup RS_n$, in which $RS_i \subseteq RS$ is a set of resource services invoked by task t_i.

When a workflow is started, a resources service sequence is formed. According to the analysis in last section, RSC is defined as follows.

Definition 2: RSC. For any workflow *Wf*, a resources service chain, denoted as $RSC = < r_1, r_2, \ldots, r_n >$, where r_i is a resource service of R_i, $r_i \in R_i$, and R_i is a CRS invoked by a task t_i, $t_i \in Task$, $1 < n <= N$, where N is maximum number of workflow from the start task to the end task.

For example as shown in Fig. 2, given a workflow, *wf*, using the notation above, we have a 4-tuple *Wf*=(*id*, *Task*, $\prec$, *RS*) where *id=101*, *Task*=$\{t_1, t_2, t_3, t_4, t_5\}$, the set of order is $\{t_1 \prec t_2, t_2 \prec t_3, t_3 \prec t_5, t_2 \prec t_4, t_4 \prec t_5\}$, and *RS*=$\{r_1, r_2, r_3, r_4, r_5\}$ in which $R_1 = \{r_1, r_2\}$, $R_2 = \{r_1, r_3, r_5\}$, $R_3 = \{r_3, r_4, r_5\}$. $RSC_1 = < r_1, r_3, r_4, r_5 >$ is an RSC, in which $r_1 \in R_1$, $r_3 \in R_2$ and $r_5 \in R_3$. R_1, R_2 and R_3 are all CRSs.

Given a workflow, the problem we are interested in is to find the optimal RSCs. The optimal RSC is actually acceptable RSC to best serve our purpose of resource service selection and recommendation, as there is some kind of dependency between resource services in the sequence. We call this problem the RSCC.

To solve the problem RSCC, CRS should be initial data source which is used to obtain optimal RSCs. This is for the reason that all resource servers in CRS are invoked by a common task and there exists a higher dependency between them than other resource services.

For example as shown in Fig. 2, there exists dependency between resource services r_1 and r_2 in CRS R_1 because they are invoked by task t_1 in common. However, there exists higher dependency between resource service r_3 and r_5 in CRS R_2 or R_3 than those in CRS R_1 because they are invoked by task t_2 and t_5 in common. Also according to workflow, there are two temporal orders between r_3 and r_5, as a result, two RSCs $< r_3, r_5 >$ and $< r_5, r_3 >$ can be obtained. The RSCC is to find the extended RSCs of $< r_3, r_5 >$ and $< r_5, r_3 >$, such as $< r_3, r_4, r_5 >$, so as to improve the resource utilization.

C. COMPOSITION OF RSC STRATEGY

The composition of RSC strategy can be divided into two stages: Building initial RSCs - At this stage, we resolve initial CRSs based on dependency, and then obtain initial RSCs according to workflow.

Optimization of extended RSC - At this stage, we optimize the extended RSC by evolutionary algorithm. The detail of this stage will be introduced in next section.

RSC is the sequence of resource services invoked by tasks, as a full sequence, can be obtained from workflow initially. In Fig. 2, the task t_3 and t_4 is executed in parallel mode, so there are 36 RSCs all together, including $< r_1, r_3, r_4, r_3 >$, $< r_2, r_3, r_5, r_4 >$, etc. However, RSC that needs to be composed is not necessarily a full sequence which is initially obtained from workflow, because that should depend on if these resource services in the RSC are usually served some common tasks. This task-related dependency between these resource services is useful for discovering their principles of usage in some fields. In order to resolve the task-related dependency between resource services in a CRS, we use the number of tasks that a CRS serves is to represent the strength of them.

Let $R = \{R_1, \ldots, R_m\}$ be a superset in which R_i denotes the CRS invoked by task t_i of workflow, $R.dep$=cnt(R', R) be a degree of dependency, which is denoted as a function to calculate the number satisfied $R' \subseteq R_i$, $i = 1, \ldots, m$. In addition R_i, degree of dependency of its subsets should be also considered, so iterations is needed until only one subset is left. Let LR=$\{LR_1,\ldots, LR_s\}$ be the superset during current iteration, NR=$\{NR_1,\ldots, NR_s\}$ be the superset during next iteration, $LR.N$ *and* $NR.N$ be their amount of sets respectively. A statistical algorithm is proposed to calculate all degrees of dependency of CRSs and their subsets. The pseudo code of the algorithm *CRSDep* is provided in Algorithm 1.

In Fig. 2, the initial set of CRS is $R = \{R_1, R_2, R_3\}$. The final set of CRS and their degrees of dependency are $R^* = \{R_1, R_2, R_3, \{r_3, r_5\}\}$ and (1, 1, 1, 2) respectively after being calculated by algorithm *CRSDep*. The set $\{r_3, r_5\}$ is a new CRS with a higher degree of dependency.

Next, based on the set of CRS, initial RSCs can be obtained by just finding the flow between resource services in CRS directly according to workflow. For example, if all of the initial CRSs with different degrees of dependency are considered, 10 RSCs will be obtained, as shown in Fig. 2. If only higher degrees of dependency are selected, $< r_3, r_5 >$ and $< r_5, r_3 >$ will be selected as initial RSCs.

Algorithm 1 CRSDep

Input: $R = \{R_1, \ldots, R_m\}$, set of CRS
Output: $R^* = \{R_1, \ldots, R_n\}$, the updated set of CRS

```
For(i =1 to m)
  R_i.dep = 1, LR ← R_i, R* ← R_i //initialize R* with R_i
End for
k = 0, LR.N= m
while(LR.N>1 )
  For(i = 1 to LR.N)
    For(j = 1 to LR.N)
        NR_k ← LR_i ∩ LR_j // calculate intersection
        if (NR_k = Ø ) then // if intersection exists
NR_k.dep ← cnt(NR_k, LR) // calculate the number of intersections
        R* ← NR_k, k++    //save for next iteration.
      End if
    End for
  End for
  k ← 0, LR ← Ø, LR ← NR // ready for next iteration
End while
return R*
```

III. RSCRSCCE- EVOLUTIONARY ALGORITHM FOR RSCC

The initial RSC obtained at the first stage of RSC composition strategy represents sequence in which the resource services are most closely associated with each other. This paper aims at improving the resource selection efficiency and utilization in CMfg. However, as resource services are invoked in workflow frequently and repeatedly, it is necessary to extend the initial RSC. The extended RSC should be optimal but not infinitely, and under control but not arbitrarily. In view of the multiple optimal solutions, inspired by the notion of natural evolution, we try to design a new intelligent algorithm for it, which is based on evolutionary algorithm.

The second stage of composition of RSC strategy, namely optimization of extended RSC will be introduced in this section.

A. REVIEW OF EVOLUTIONARY ALGORITHM

An evolutionary computation (EA) uses mechanisms inspired by biological evolution, such as reproduction, mutation, recombination, and selection, to solve combinatorial optimization problems. GA-based (Genetic Algorithm-based) strategies have been recognized as efficient solutions for heuristically solving complex and intractable optimization problems across various domains. Relevant researches include resource-constrained project scheduling problem [26], [27], ranking problem [28], task scheduling in cloud computing [29], and software-defined networks [30], etc. In GA, a suitable chromosome representation, variable-length or fixed-length, is needed to encode potential solutions to the

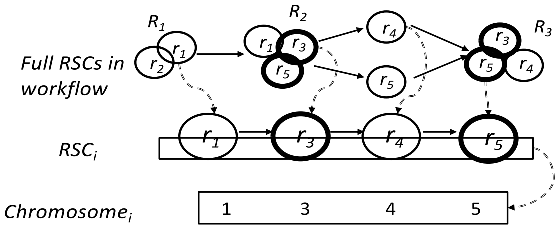


FIGURE 3. Schematic representation of chromosome.

problem that the GA algorithm is trying to solve. A general GA first initializes and evaluates the population to select the best fitting chromosomes. It then applies crossover and mutation operators to generate and evaluate the new offspring [31]. A fitness function is also required to evaluate how well a candidate solution performs. If the fitness function is defined imprecisely, the GA may be unable to find a solution to the problem.

B. CCHROMOSOME REPRESENTATION OF RSC

One of the key issues in RSCCE with GA is finding a suitable chromosome representation as potential solution to a problem. The detailed way to encode solutions depends on the nature of the problem. For the considered RSCC problem, N different types of resource services invoked by a workflow constitute a finite set RS. An RSC $< r_1, r_2, \ldots, r_n >$ is composed of resource service set $\{r_1, r_2, \ldots, r_n\}$, which is a subset of RS, where $n \leq N$.

Now, suppose that the maximum length of workflow execution paths, namely the number of tasks from start task to end task in workflows, is L. The maximum length of RSCs associated with a workflow is less than or equal to L. Therefore, an RSC $< r_1, r_2, \ldots, r_L >$ can be represented as a solution chromosome, and each chromosome is made of L genes, in which the i-th gene can exactly be represented as the i-th resource service r_i. Such integer representation of the chromosome is suitable for the RSCC problem, so that a search space of N dimension can be set up for RSCs.

A schematic representation of the chromosome is shown in Fig. 3. There are 5 types of resource service invoked by workflow. The maximum length of the workflow execution path is 4. Of all the RSCs, the RSC $< r_1, r_3, r_4, r_5 >$ can be represented by chromosome (1345).

C. FITNESS FUNCTION

The fitness function $fitness(c_i)$ measures to what extent the candidate solution satisfies some criteria. The basic genetic operators are selection, crossover and mutation. In the selection process, an individual is selected for the next population with the number of copies proportional to the fitness value [31].

Given an initial RSC RSC_i, if it is spaced by other resource services and formed a new sequence RSC'_i, we call the new generated sequence RSC'_i an extended RSC (exRSC) of RSC_i.

Algorithm 2 RSCCE

Input: rsc=$< r_1, \ldots, r_m >$, an initial RSC
  RS, resource service set invoked by a workflow
  $fullRSC$, a set of full RSCs associated with a workflow
Output: $RSC' = \{RSC'_1, \ldots, RSC'_s\}$, exRSC of rsc

```
P ← Ø;
initPop (rsc, RS);
calFitness ();
while(gen<MAXGEN) {
    gen ++, PS ← Ø;
    PS ← select(P);
    crossover ();
    mutate ();
    P ← PS;
    calFitness ();
End while
```

In order to select better chromosomes and obtain optimal exRSC, fitness function is defined as follows:

$$f_j = \frac{\sum_{i=1}^{n-1} d_{i,i+1} \times \delta^{d_{i,i+1}-1}}{n}, \quad (n \leq L, 0 \leq j < PSIZE), \qquad (1)$$

where $d_{i,\,i+1}$ is the distance between r_i and r_{i+1}, measured by interval length between r_i and r_{j+1}, $PSIZE$ is the number of population, n is the length of exRSC, r_i and r_{i+1} are the two adjacent resource services in exRSC, L is the maximum length of workflow execution paths.

If any two adjacent resource services r_i and r_{i+1} are spaced by other k resources $< r'_i, \ldots, r'_{i+k} >$, the interval length between r_i and r_{j+1} is denoted as $d_{i,\,i+1}$. A larger value of $d_{i,\,i+1}$ indicates more dissimilar between an RSC and its extRSC. In addition, the similar degree depends on the number of intervals, so in (1), all intervals value should be summed up. Based on previous research [32], the similar degree can decrease exponentially with the increasing value $d_{i,\,i+1}$ in an interval $< r_i, r_{i+1} >$. Therefore, we set the interval length of adjacent resource services r_i, r_{i+1} to 1, namely $d_{i,\,i+1} = 1$, take $\delta = 0.4$ as base, $d_{i,\,i+1} - 1$ as exponent.

D. EVOLUTIONARY ALGORITHM FOR RSCC

The genetic algorithm for RSCC finds the optimal solutions, the extRSCs of initial RSCs. The pseudo code of the algorithm *RSCCE* is provided in Algorithm 2.

1) INITIAL POPULATION AND CONSTRAINTS

The initial population consists of $PSIZE$ randomly generated individuals, where $PSIZE$ is the population size, as a control parameter. Firstly, a chromosome is generated by selecting L resource services randomly from the set RS, where RS is resource services invoked by a workflow. Next, n genes are selected from the chromosome, and replaced by the resource services $r_1, \ldots, r_m$ respectively in the initial RSC rsc=$< r_1, \ldots, r_m >$, where $n = |rsc|$, the length of rsc.

Algorithm 3 InitPop

Input: $rsc = < r_1, \ldots,$
$r_m >$, an initial RSC
RS, resource service set invoked by a workflow
Output: $RSC' = \{RSC'_1, \ldots, RSC'_s\}$, exRSC of rsc

while($|P| < PSIZE$) {
  c_i = random (RS); //generate each chromosome randomly according to RS
  c_i = generate (c_i,rsc); // select $|rsc|$ genes randomly, replace them with resource services in rsc in its sequence.
  if(constraint(c_i) == TRUE) then add c_i to P;
End while

Algorithm 4 Select Operator

Input: a population P
Output: next generation population PS

  $F = 0$;
for($i = 0$; $i < PSIZE$; $i++$)$F = F + f_i$;
  $p_0 = f_0 / F$;//calculate the selection probability of the first individual
  for($i = 0$; $i < PSIZE$; $i++$)$p_{i+1} = f_i / F + p_i$; // the selection probability of the $i-th$ individual
  for($i = 0$; $i < PSIZE$; $i++$) {
    find j where $p_j <$rand (1)$\leq p_j$+1//j is proportion of F
   add c_j to PS from P
End for

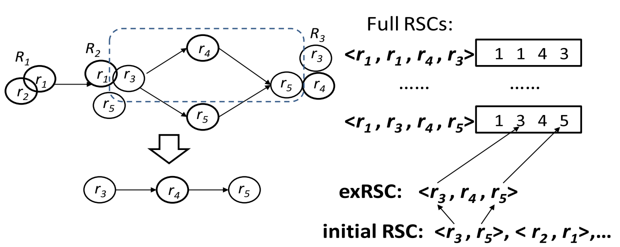


FIGURE 4. Constraints of chromosomes.

Finally, constraints must be satisfied. A detailed description of function *InitPop* is given in Algorithm 3.

As each initial RSC is composed of a sequence of task-related resource service, it is the basis to resolve. Obviously, the optimal solutions are variable-length individuals. Their lengths are no less than L, where L is the maximum length of workflow execution paths. The variable-length chromosomes should satisfy the following criteria: 1) the genes in chromosome, as resource services in an RSC, keep the same temporal orders with its initial RSC; 2) the genes in chromosome of initial RSC keep the same temporal orders with those of any full RSC, which is formed of being invoked by workflow tasks; and 3) the genes in chromosome of a full RSC keep the same temporal orders with those of the solution chromosome. The RSC $< r_3, r_4, r_5 >$ is an exRSC of initial RSC $< r_3, r_5 >$, as shown in Fig. 4, in which r_3 and r_5 keep the same temporal orders with a full RSC $< r_1, r_3, r_4, r_5 >$. The $constraint(c_i)$ in algorithm 2 is used to ensure that the optimal solutions meet the constraints above.

2) SELECTION OPERATOR

Selection operator, as an important part of genetic algorithms, follows the rule: The better fitted an individual, the larger the probability of its survival and mating [31]. Roulette-wheel selection [33] is a traditional GA selection technique, which assumes that the probability of selection is proportional to the fitness of an individual. Suppose that there are N individuals in a population, each of which is characterized by its fitness f_i, where $f_i > 0$ ($i \leq N$). The selection probability of the *i-th* individual can thus be expressed as (1).

$$p_i = f_i / \sum_{j=1}^{N} f_j, \quad (i = 1, 2, \ldots, N) \tag{2}$$

In Algorithm 2, (1) is implemented by the function *calFitness*. The algorithm of function *select* is given as Algorithm 4.

3) CROSSOVER OPERATOR

Crossover operator is used to replace some of the genes in one parent with corresponding genes of the other. In RSCC problem, a single point crossover is applied. Firstly, the cut-off point j is selected randomly to cut the chromosome into two segments, the left and the right, where $0 < j < L$, as shown in Fig. 5. Then, the genes of left segment are copied from another parent and replaced by them one by one.

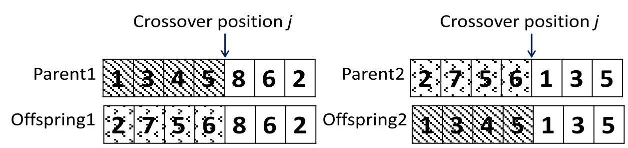


FIGURE 5. Single-point crossover operator.

The algorithm of function crossover is given as Algorithm 5.

4) MUTATION OPERATOR

The mutation operator can maintain the diversity of the population to enlarge the search space of exRSC. Firstly, a mutation position is selected randomly from a chromosome, where $0 < j < L$. Then the gene at the position j is replaced by another gene, which is a resource service represented by an integer. Suppose that the probability of mutation is pm, the algorithm of function *mutation* is given as Algorithm 6.

IV. SIMULATION AND RESULTS

A. EXPERIMENTAL SETUP

We take collaborative design and manufacturing processes of electrical apparatus as a case, to analyze our method,

Algorithm 5 Crossover Operator

Input: a population *PS*
Output: the updated *PS* after doing crossover

for($i = 0$; $i < pc*PSIZE/2$;) //*pc* is the probability of crossover
  select c_m, c_n pair from *PS* at random;// c_m and c_n are the two individuals.
  find crossover point j ($0 < j < L$) at random;// single point
  for($k = 0$; $k < j$; $k++$) //copy left segment to another parent
  copy g_k from c_m to c'_n; //g_k is a gene
  copy g_k from c_n to c'_m;
End for
  for($k = j$; $k < L$; $k++$) // generate new offspring c'_m and c'_n
    copy g_k from c_m to c'_m;
    copy g_k from c_n to c'_n;
  End for
  if(constraint(c'_m) = TRUE and constraint (c'_n)= TRUE) then update *PS* with c'_m and c'_n; $i++$;
  End for

Algorithm 6 Mutation Operator

Input: a population *PS*
Output: the updated *PS* after doing mutation

for(i =0; i <$pm*PSIZE$;) {//*pm* is the probability of mutation
  select c_m from *P* at random;// cm is the individual.
  find mutation point j ($0 < j < L$) at random;
  replace g_j with rand(|RS |), generate a new chromosome c'_m //generate a integer, to replace the gene at position j
  if(constraint(c_m) = TRUE then
    update *PS* with c'_m;
    $i++$;
  End if
End for

as shown in Fig. 1. There are 5 business processes in this case, including product designing, hardware and machine manufacturing, product assembling and parts supplying. Product designing needs four different professions to work cooperatively, which are hardware design, software design, machine design and power design.

The business processes need invoke resource services, including hardware resource, human resources and technology resources, as shown in Table 1.

A resource service graph is formed along with the execution of the workflow, featured by collaboration, as shown in Fig. 6. In the graph, there are 912 full RSCs all together.

The simulation experiment has two steps: 1) setting up the initial set of RSCs according to workflow model by

TABLE 1. Tasks and resource services invoked by tasks.

T	Resources	T	Resource
t_3	r_1: assembling data and price r_7:hardware data and price r_{10}:machine part data and price r_{13}:part standards and price	t_6	r_1: assembling data and price r_3: machine designer r_{10}: machine part data and price r_{15}:machine part configuration
t_{13}	r_{18}:part	t_5	r_2: product design scheme r_4: software designer
t_4	r_2:product design scheme r_5:hardware designer r_7:hardware data and price	t_7	r_2: product design scheme r_6:power designer r_{16}: power configuration
t_{20}	r_2: product design scheme r_3: machine designer r_4: software designer r_5: hardware designer r_6: power designer	t_{15}	r_{14}: machine part scheme r_{17}:power supply scheme
t_{12}	r_3: machine designer r_9:machine manufacturing scheme	t_{10}	r_{18}:part
t_{19}	r_{18}:part r_{11}: machine part r_{12}: hardware	t_9	r_5: hardware designer r_8: hardware manufacturing scheme

[a]T = Task

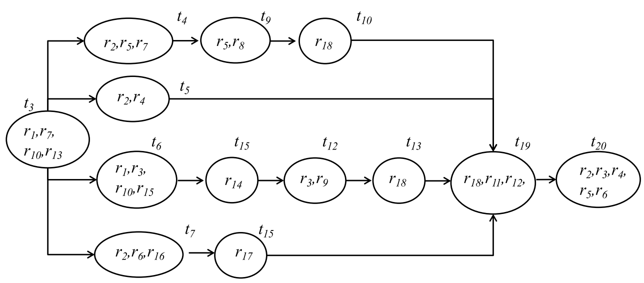


FIGURE 6. Resource service graph of workflow.

Algorithm CRSDep; 2) resolving optimal revolutions using the RSCCE algorithm.

Steps 1 *Setting Up the Initial Set of RSCs:* According to the task-related dependency between resource services in workflow model, as shown in Fig. 6, using Algorithm CRSDep, the initial RSCs are $<\{r_1, r_{10}, r_{13}\}, r_7>$, $<r_2, r_5>$, $<r_5, \{r_2, r_3, r_4\}>$, $<r_3, \{r_2, r_4, r_5\}>$, $<r_4, \{r_2, r_3\}>$, $<r_{16}, r_6>$, $<r_2, \{r_3, r_4\}>$, $<r_{18}, \{r_{11}, r_{12}\}>$ and $<r_4, r_5>$.

Steps 2 *Resolving Optimal Revolutions Using the RSCCE Algorithm:* Based on the set of RSCs above, the RSCC problem for the collaborative processes can be solved using the RSCCE algorithm. The parameters in the algorithms are set as follows: (1) the size of the population is 15; (2) the length of chromosome is 7; (3) the termination condition is 200 generations reached; (4) the crossover probability is 0.8; (5) the mutation probability is 0.1, and (6) the value of fitness is from 0.6 to 1.

Using these parameters, there are a larger number of optimal solutions. For the purpose of resolving optimal solutions, in each generation, RSCCE will remove duplicate optimal solutions from current population and only retains new

TABLE 2. Result of RSCCE for optimal revolutions.

Chromosome	Fitness	Chromosome	Fitness
1 7	1.0	18 12	1.0
10 7	1.0	4 11 5	0.9
13 7	1.0	4 18 5	0.9
2 11 5	0.9	4 12 5	0.9
2 17 18 5	0.6075	5 18 12 2	0.6075
2 5	1.0	5 18 18 3	0.6075
2 17 12 5	0.6075	5 18 12 3	0.6075
5 18 18 2	0.6075	5 18 11 3	0.6075
5 18 11 2	0.6075	5 18 11 4	0.6075
3 18 12 2	0.6075	5 18 18 4	0.6075
3 18 11 4	0.6075	5 18 12 4	0.6075
3 18 12 4	0.6075	3 18 11 2	0.6075
3 18 18 4	0.6075	3 18 18 2	0.6075
3 18 11 5	0.6075	4 18 2	0.9
3 18 18 5	0.6075	4 11 3	0.9
3 18 12 5	0.6075	4 12 3	0.9
4 11 2	0.9	4 18 3	0.9
4 12 2	0.9	16 17 12 6	0.6075
2 12 3	0.9	16 17 18 6	0.6075
2 11 4	0.9	16 17 11 6	0.6075
2 18 4	0.9	2 11 3	0.9
2 12 4	0.9	2 18 3	0.9
18 11	1.0	-	-

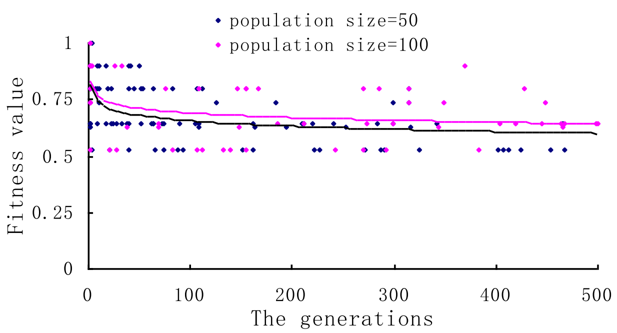


FIGURE 7. Two experimental results for population size is 50 and 100.

optimal solutions which are different from those in previous generations. In the following generations, the number of new optimal solutions remains steady at approximately between 0 and 2. There are 45 optimal solutions, as shown in Table 2.

B. RESULTS AND DISCUSSIONS

To illustrate the trend of evolution, parameters are adjusted, the size of the population set to 50 and 100 respectively, the maximum generations set to 500. There are 94 and 63 optimal revolutions respectively. The two similar trend curves indicate that, as shown in Fig. 7, RSCCE has steadiness in finding optimal solutions.

In some generations, there is no optimal solution to be resolved. One of the reasons is the constraint violation. The 3 constraints of RSCCE have been introduced in section III, of which the most important criteria is "keep the same temporal order with its initial RSC", so that lots of chromosomes are removed in every generation. That is also the main bottleneck of RSCCE performance.

To illustrate the power of RSCCE, we still take the same case *collaborative design and manufacturing processes of electrical apparatus*, to compare this algorithm with the previously proposed RSCCA algorithms [6]. We use the same strategy, i.e. the task-related dependency between resource services, to resolve initial set of RSCs. Based on the same initial set of RSCs, we first compare the number and the precision of optimal solutions between RSCCA and RSCCE. The initial RSCs are $rsc_1 =< \{r_1, r_{10}, r_{13}\}, r_7 >$, $rsc_2 =< r_2, r_5 >$, $rsc_3 =< r_5, \{r_2, r_3, r_4\}>$, $rsc_4 =< r_3, \{r_2, r_4, r_5\}>$, $rsc_5 =< r_4, \{r_2, r_3\}>$, $rsc_6 =< r_{16}, r_6 >$, $rsc_7 =< r_2, \{r_3, r_4\}>$, $rsc_8 =< r_{18}, \{r_{11}, r_{12}\}>$ and $rsc_9 =< r_4, r_5 >$. The quantity of optimal solutions is compared, as shown in Fig. 8. The optimal solutions are candidate RSCs which will be provided to the business process. Therefore, a smaller candidate set of RSCs can contribute more efficient resource service selection to a business process.

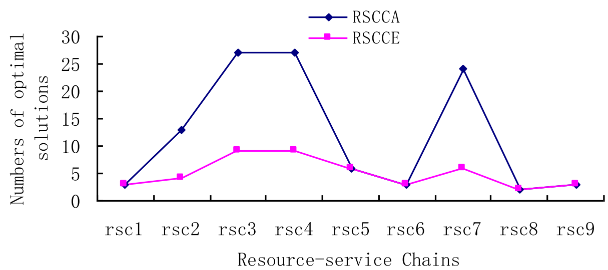


FIGURE 8. Comparison of candidate RSCs quantity between RSCCA and RSCCE.

However, in addition to requirement of quantity, candidate set of RSCs should be as precise as possible because a more precise candidate set of RSCs can bring more efficient to a business process. Therefore, we use formula (3) [6] to calculate the distance to measure precision, between the optimal solutions and their initial set of RSCs. In formula (3), $|RSC|$ is the length of RSC. The result of comparison is shown in Fig. 9. By comparison, the optimal solutions RSCCE resolve has a significant advantage.

$$\mathrm{dis}(\mathrm{RSC}, \mathrm{RSC}'_i) = (\left|\mathrm{RSC}'_i\right| - |\mathrm{RSC}|)/\left|\mathrm{RSC}'_i\right| \tag{3}$$

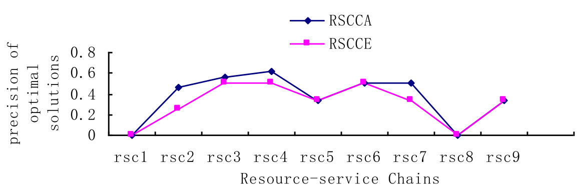


FIGURE 9. Comparison of candidate RSCs precision between RSCCA and RSCCE.

V. CONCLUSION

Composition of resource service chain is an important problem in CMfg system. In this paper, an approach RSCCE is proposed to improve the efficient of resource-service selection. Steadiness is the advantage of RSCCE. In the optimal solutions, the candidate RSCs are better if they are more similar with their initial RSCs than in other methods. These candidate RSCs have more opportunities to be chosen. A recommended future work focuses on clustering algorithm to deal with large scale data. Though some relevant researches for large scale data [34]–[36] and time series data [37] have been carried out, fast clustering algorithm applied to distributed cloud manufacturing system should be paid more attention. This is helpful to improve practicality of resource usage.

REFERENCES

[1] H. Li, M. Liang, and T. He, "Optimizing the composition of a resource service chain with interorganizational collaboration," *IEEE Trans. Ind. Informat.*, vol. 13, no. 3, pp. 1152–1161, Jun. 2017.

[2] F. Tao, Y. Zuo, L. Da Xu, and L. Zhang, "IoT-based intelligent perception and access of manufacturing resource toward cloud manufacturing," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1547–1557, May 2014.

[3] G. D. Jules, M. Saadat, and S. Saeidlou, "Holonic ontology and interaction protocol for manufacturing network organization," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 5, pp. 819–830, May 2015.

[4] F. Tao, Y. Laili, L. Xu, and L. Zhang, "FC-PACO-RM: A parallel method for service composition optimal-selection in cloud manufacturing system," *IEEE Trans. Ind. Informat.*, vol. 9, no. 4, pp. 2023–2033, Nov. 2013.

[5] J. Andrade, J. Ares, R. Garcia, M. A. Martinez, J. Pazos, and S. Suarez, "A game theory based approach for building Holonic virtual enterprises," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 2, pp. 291–302, Feb. 2015.

[6] H. Li, K. Chan, M. Liang, and X. Luo, "Composition of resource-service chain for cloud manufacturing," *IEEE Trans. Ind. Informat.*, vol. 12, no. 1, pp. 211–219, Feb. 2016.

[7] F. Tao, Y. Cheng, L. Da Xu, L. Zhang, and B. Hu Li, "CCIoT-CMfg: Cloud computing and Internet of Things-based cloud manufacturing service system," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1435–1442, May 2014.

[8] X. Xu, "From cloud computing to cloud manufacturing," *Robot. Comput.-Integr. Manuf.*, vol. 28, no. 1, pp. 75–86, Feb. 2012.

[9] K. M. Sim, "Agent-based cloud computing," *IEEE Trans. Serv. Comput.*, vol. 5, no. 4, pp. 564–577, 4th Quart., 2012.

[10] M. Ruta, F. Scioscia, G. Loseto, and E. Di Sciascio, "Semantic-based resource discovery and orchestration in home and building automation: A multi-agent approach," *IEEE Trans. Ind. Informat.*, vol. 10, no. 1, pp. 730–741, Feb. 2014.

[11] F. Ahmad and A. Sarkar, "Analysis of dynamic Web services: Towards efficient discovery in cloud," *Malaysian J. Comput. Sci.*, vol. 29, no. 3, pp. 156–178, Nov. 2017.

[12] S. N. Han, G. M. Lee, and N. Crespi, "Semantic context-aware service composition for building automation system," *IEEE Trans. Ind. Informat.*, vol. 10, no. 1, pp. 752–761, Feb. 2014.

[13] B. Xu, L. Da Xu, H. Cai, C. Xie, J. Hu, and F. Bu, "Ubiquitous data accessing method in IoT-based information system for emergency medical services," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1578–1586, May 2014.

[14] Y. Xia, X. Luo, J. Li, and Q. Zhu, "A Petri-net-based approach to reliability determination of ontology-based service compositions," *IEEE Trans. Syst., Man, Cybern.*, vol. 43, no. 5, pp. 1240–1247, Sep. 2013.

[15] H. Li and Y. Gao, "Mining QoS benchmark of resource-service chain for collaborative tasks," *Int. J. Internet Manuf. Serv.*, vol. 5, nos. 2–3, p. 121, 2018.

[16] H. B. Li and X. Y. Lei, "Identifying key feature sequence of resource services for collaborative task," *Comput. Integr. Manuf.*, vol. 23, no. 12, pp. 2571–2582, 2017.

[17] S. Zhang and H.-B. Li, "Resource selection method based on workflow in cloud manufacturing," *Comput. Integr. Manuf.*, vol. 21, no. 3, pp. 831–839, Mar. 2015.

[18] R. J. Rodriguez, J. Julvez, and J. Merseguer, "On the performance estimation and resource optimization in process Petri nets," *IEEE Trans. Syst., Man, Cybern.*, vol. 43, no. 6, pp. 1385–1398, Nov. 2013.

[19] S. C. Geyik, B. K. Szymanski, and P. Zerfos, "Robust dynamic service composition in sensor networks," *IEEE Trans. Serv. Comput.*, vol. 6, no. 4, pp. 560–572, Oct. 2013.

[20] S.-E. Tbahriti, C. Ghedira, B. Medjahed, and M. Mrissa, "Privacy-enhanced Web service composition," *IEEE Trans. Serv. Comput.*, vol. 7, no. 2, pp. 210–222, Apr./Jun. 2014.

[21] O. Bushehrian, "Model-based service selection for reliable service access in manet," *Malaysian J. Comput. Sci.*, vol. 27, no. 4, pp. 294–306, 2014.

[22] H. Li and T. He, "Selecting key feature sequence of resource services in industrial Internet of Things," *IEEE Access*, vol. 6, pp. 72152–72162, 2018.

[23] H. Li, J. Tong, S. Weng, X. Dong, and T. He, "Detecting a business anomaly based on QoS benchmarks of resource-service chains for collaborative tasks in the IoT," *IEEE Access*, vol. 7, pp. 165509–165519, 2019.

[24] H. Li and J. Tong, "A novel clustering algorithm for time-series data based on precise correlation coefficient matching in the IoT," *Math. Biosci. Eng.*, vol. 16, no. 6, pp. 6654–6671, 2019.

[25] H. B. Li, "Approach to multi-granularity resource composition based on workflow in cloud manufacturing," *Comput. Integr. Manuf.*, vol. 19, no. 1, pp. 210–216, 2013.

[26] X. Sun, S. Guo, J. Guo, and B. Du, "A hybrid multi-objective evolutionary algorithm with heuristic adjustment strategies and variable neighbor-hood search for flexible job-shop scheduling problem considering flexible rest time," *IEEE Access*, vol. 7, pp. 157003–157018, 2019.

[27] M. A. Dulebenets, "A comprehensive evaluation of weak and strong mutation mechanisms in evolutionary algorithms for truck scheduling at cross-docking terminals," *IEEE Access*, vol. 6, pp. 65635–65650, 2018.

[28] J.-Y. Yeh and J.-Y. Lin, "Learning ranking functions for information retrieval using layered multi-population genetic programming," *Malaysian J. Comput. Sci.*, vol. 30, no. 1, pp. 27–47, Nov. 2017.

[29] F. Zhang, J. Cao, K. Hwang, K. Li, and S. U. Khan, "Adaptive workflow scheduling on cloud computing platforms with iterativeordinal optimization," *IEEE Trans. Cloud Comput.*, vol. 3, no. 2, pp. 156–168, Apr. 2015.

[30] W. Chen, H. Chen, Q. Guan, F. Ji, and B. Guo, "Evolutionary sleep scheduling in software-defined networks," *IEEE Access*, vol. 6, pp. 29541–29550, 2018.

[31] C.-H. Chen, T.-K. Liu, and J.-H. Chou, "A novel crowding genetic algorithm and its applications to manufacturing robots," *IEEE Trans. Ind. Informat.*, vol. 10, no. 3, pp. 1705–1716, Aug. 2014.

[32] H.-B. Li, D.-C. Zhan, and X.-F. Xu, "Dynamic component prefetching method for workflow management system," *Chin. J. Comput.*, vol. 35, no. 5, pp. 1038–1045, Nov. 2012.

[33] D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA, USA: Addison-Wesley, Jan. 1989.

[34] Y. Chen, L. Zhou, S. Pei, Z. Yu, Y. Chen, X. Liu, J. Du, and N. Xiong, "KNN-BLOCK DBSCAN: Fast clustering for large-scale data," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published, doi: 10.1109/tsmc.2019.2956527.

[35] Y. Chen, X. Hu, W. Fan, L. Shen, Z. Zhang, X. Liu, J. Du, H. Li, Y. Chen, and H. Li, "Fast density peak clustering for large scale data based on kNN," *Knowl.-Based Syst.*, vol. 187, Jan. 2020, Art. no. 104824.

[36] Y. Chen, S. Tang, N. Bouguila, C. Wang, J. Du, and H. Li, "A fast clustering algorithm based on pruning unnecessary distance computations in DBSCAN for high-dimensional data," *Pattern Recognit.*, vol. 83, pp. 375–387, Nov. 2018.

[37] S. Pei, T. Shen, X. Wang, C. Gu, Z. Ning, X. Ye, and N. Xiong, "3DACN: 3D augmented convolutional network for time series data," *Inf. Sci.*, vol. 513, pp. 17–29, Mar. 2020.

HAIBO LI received the Ph.D. degree from the School of Computer Science and Technology, Harbin Institute of Technology, in 2008. He worked as a Research Scholar at Hong Kong Polytechnic University, Hong Kong, from September 2014 to March 2015. He is currently a Professor with the College of Computer Science and Technology, Huaqiao University, Xiamen, China. He is author or coauthor of over 40 publications and has received several awards for his research. His published works appear in several refereed journals, including *Journal of Software, Chinese Journal of Computers*, and *Computer Integrated Manufacturing System*. His research interests are in the areas of service-oriented manufacturing system, such as cloud manufacturing and manufacturing service management, and intelligent optimization theory and algorithm.

SHAOYUAN WENG received the bachelor's degree from the Quanzhou University of Information Engineering, in 2018. She is currently pursuing the M.S. degree with the College of Computer Science and Technology, Huaqiao University, Xiamen, China. Her main research interests are in the areas of analysis of data mining and collaborative task.

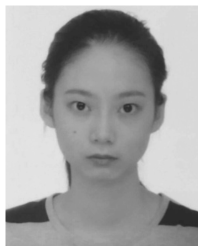
JUNCHENG TONG received the bachelor's degree from Tianjin Agricultural University, in 2017. He is currently pursuing the M.S. degree with the College of Computer Science and Technology, Huaqiao University, Xiamen, China. His main research interests are in the areas of service science and big data analytics.

TING HE was born in 1972. He received the B.S. and M.S. degrees in mechanical design and manufacturing and management engineering from the Harbin Institute of Technology, in 1995 and 1997, respectively, and the Ph.D. degree from the Department of Mechatronical Engineering, Harbin Institute of Technology, in 2000.

He worked at the School of Computer Science and Technology, Harbin Institute of Technology, from 2000 to 2016. He is currently a Professor with the College of Computer Science and Technology, Huaqiao University. He is currently exploring the data-enabled innovation of business model and intelligent / service oriented manufacturing. His main research interest includes smart computing and services.

WENYUN CHEN is currently pursuing the B.S. degree with the College of Computer Science and Technology, Huaqiao University, Xiamen, China. Her main research interests are in the areas of analysis of data mining and collaborative task.

MENGMENG SUN is currently pursuing the B.S. degree with the College of Computer Science and Technology, Huaqiao University, Xiamen, China. Her main research interests are in the areas of analysis of data mining and collaborative task.



YINGTONG SHEN is currently pursuing the B.S. degree with the College of Computer Science and Technology, Huaqiao University, Xiamen, China. Her main research interests are in the areas of analysis of data mining and collaborative task.