International Journal of Innovative
Computing, Information and Control
Volume **14**, Number **3**, June **2018**

ICIC International ©2018 ISSN 1349-4198
pp. **1057–1077**

LOCATION SENSITIVE MULTI-TASK ORIENTED SERVICE COMPOSITION FOR CYBER PHYSICAL SYSTEMS

Yuan Sun, Xingshe Zhou and Gang Yang

School of Computer Science and Engineering
Northwestern Polytechnical University
No. 127, West Youyi Road, Beilin District, Xi'an 710072, P. R. China
sunyuan@mail.nwpu.edu.cn; { zhouxs; yeungg }@nwpu.edu.cn

Received August 2017; revised December 2017

Abstract. *As a core technology to enable service-based cyber physical systems (CPSs), CPS service composition has received a lot of attention from researchers. However, existing studies on it mostly aim to solve single task oriented CPS service composition problem, and the multi-tasking requirements of CPSs are not considered. In fact, there are quite a few situations where multiple tasks need to be completed as soon as possible, such as smart emergency response, and smart manufacturing. In addition, locations have much influence on the execution time of CPS services because before CPS services begin to work for tasks, the required physical entities must move or be transported to the predefined locations. In this paper we investigate the CPS service composition problem in which these two characteristics are both considered. To address it, we not only need to select appropriate CPS services for each task, but also need to arrange the invocation sequence for each selected CPS service. The final goal is to find a service selection and invocation scheme that can optimize the makespan as much as possible. A heuristic approach based on an improved quantum genetic algorithm is designed to handle it. Experimental results show that the proposed method can find the superior schemes. Compared with the schemes given by other methods, our schemes can achieve lower makespan under various situations, such as different number of CPS services and tasks, and different types of spatial distribution.*

Keywords: Cyber physical systems (CPSs), Service composition, Quantum genetic algorithm, Location sensitivity, Multiple tasks

1. **Introduction.** Recent years have witnessed rapid development and continuous integration of such technologies as embedded computing, wireless sensor network and networked control. These progresses make a large number of physical entities integrated into the cyber world, thus leading to the emergence of cyber physical systems (CPSs) [1, 2, 3]. The representative CPS application scenarios include smart emergency response, swarms of UAVs (unmanned aerial vehicles), and smart manufacturing.

In CPSs, the cyber world and the physical world are tightly integrated [4, 5]. Physical entities (PEs), e.g., sensors, actuators, robots, and drones, and cyber entities (CEs), e.g., software modules for analysis, control and decision making, mutually collaborate to conduct sophisticated operations [4]. The events in the physical world are first reflected in the cyber world, where they are used to make control decisions that are returned to the physical world to adjust physical processes [5]. Moreover, in order to enhance this collaboration, in recently emerging cloud-based CPSs [6, 7, 8], expensive computational operations can even be performed on clouds. This tight integration makes researchers realize that treating cyber entities and physical entities as very separate concepts is ineffective to manage and leverage them. As a sequence, service computing technologies have been gradually applied in CPSs [9, 10, 11, 12, 13].

In service-based CPSs, a set of CEs and PEs that closely work together are abstracted into a CPS service. CPS services are treated as basic structural units of application tasks that arise now or may arise in the future. A core technology to enable service-based CPSs is CPS service composition, which can compose a number of CPS services to complete application tasks.

There have been many studies on CPS service composition [14, 15, 16, 17]. Although these works are well done, unfortunately, they are mostly single task oriented, and the multi-tasking requirements of CPSs are not considered. The multi-tasking requirements are closely related to the exclusivity of CPS services. In general, a CPS service can only work for one task at a time, because a CPS service is integrated with operations of one or more physical entities, and a physical entity can usually perform only one operation at a time. If there are multiple tasks that need to be completed as soon as possible, CPS service composition will be very different from traditional service composition. For CPS service composition, the service selection conflicts might happen because one CPS service might be selected to work for more than one task. Nevertheless, it is not the same case for traditional service composition, because software services, especially the ones deployed on clouds, are designed to handle concurrent invocations at a very short time [18].

In addition, the location sensitivity of CPS services also needs to be considered. In existing works [12, 14, 16, 17], locations are mostly treated as preconditions or effects of CPS services. Actually, the location sensitivity of CPS services is also embodied in other respects. On the one hand, because tasks in CPSs are often location dependent, before CPS services are really executed to work for tasks, the required PEs need to move to the designated locations. In this sense, the execution time of a CPS service usually includes two parts: the moving time, during which the required PE moves to the designated location, and the operating time, during which the CPS service is really executed. On the other hand, under certain circumstances, the PEs required by some CPS services are incapable of moving or do not have enough energy for moving. At this point, these CPS services must employ other CPS services (called transportation services) to transport their PEs.

In this paper, we focus on the location sensitive multi-task oriented CPS service composition (LMSC) problem. To illustrate the LMSC problem, we introduce a motivation example, a smart emergency response scenario where two valuable objects (at locations A and B, respectively) in a disaster site are required to be transfered to a safe place (denoted by S) quickly (see Figure 1). We assume that the environment in the disaster site, such as the positions of the objects, obstacles and fires, are well known, and according

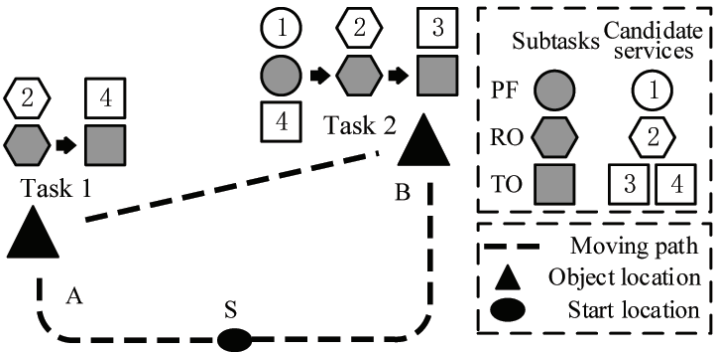

FIGURE 1. A simple smart emergency response scenario where two valuable objects are required to be transfered to a safe place quickly (PF – Putting out fires. RO – Removing obstacles. TO – Transporting objects.)

to this information, the operating steps of the two transferring tasks (denoted by tasks 1 and 2, respectively) can be obtained. There are four CPS services, each of which can only accomplish one operating step, e.g., putting out fires, removing obstacles, and transporting objects. Both the first and second CPS services require a mobile robot, and the robot used by the first CPS service does not have enough energy for moving. The last two CPS services are transportation services, and each of them uses a claw-handed drone to transport objects.

To address it, we need to find five CPS services to accomplish the total five subtasks. However, there are only four CPS services available, so it is inevitable that certain CPS service will be selected to work for more than one subtask. If the first CPS service is selected, a transportation service is required to help it, because the robot used by the first CPS service does not have enough energy for moving.

In general, three sub-problems must be solved simultaneously to deal with the LM-SC problem: (1) selecting a set of CPS services for each task, (2) selecting appropriate transportation services for some CPS services, and (3) resolving possible service selection conflicts. The second sub-problem comes from the location sensitivity of CPS services. The last sub-problem originates from the multi-tasking requirements. To resolve the possible conflicts, we need to arrange the invocation sequence for each selected CPS service. The final goal is to find a service selection and invocation scheme that can optimize the makespan (the time for completing all tasks) as much as possible. The contributions of this paper are summarized as follows.

1) To the best of our knowledge, it is the first time that the multi-tasking requirements and the location sensitivity of CPS services are simultaneously considered in CPS service composition.

2) We formulate the location sensitive multi-task oriented CPS service composition problem in theory by a mixed integer programming model.

3) A heuristic approach based on an improved quantum genetic algorithm (called SSI-IQGA) is presented to deal with the LMSC problem. Compared to the basic quantum genetic algorithm, the improvements such as dynamic rotation angle mechanism, quantum variation and quantum crossover are integrated into SSI-IQGA to enhance the searching performance of the algorithm.

4) Simulations are conducted to evaluate the performance of the proposed method. The results indicate that the proposed method outperforms other methods under different situations.

The rest of this paper is organized as follows. Section 2 presents the related work. Section 3 provides the problem definition. Section 4 details the proposed heuristic approach based on an improved quantum genetic algorithm. Section 5 evaluates the performance of the proposed algorithm. Section 6 draws the conclusion and discusses the future work of this paper.

2. **Related Work.**

In general, before CPS service composition is conducted, an appropriate service model is usually required to accurately describe CPS services.

CPS service description refers to defining functional and non-functional properties of CPS services. Huang et al. [19] first extended the web ontology language for services (OWL-S) [20] to construct a context-sensitive resource-explicit service model. In the model, the concept of service provision constraints is proposed to define the relationships between the CPS services provided a PE. Subsequently, Huang et al. [14] proposed two new concepts, namely context precondition and context effect, to describe the special types of constraints that are related to the context of PEs. In [12], these two concepts are reserved. Besides them, Zhu et al. proposed a concept called "AppliedTo". It is used to

express the fact that after the execution of a CPS service, the context of some PE may change. Jin et al. [21] proposed a service model that can define time and space related characteristics. Wan et al. [22] presented a resource-centric CPS service model in which a resource description template is employed to model PEs.

CPS service composition is utilized to compose a number of CPS services to complete application tasks. Huang et al. [14] proposed an iterative AI planning based service composition approach. An AI planning step consists of two stages: abstract composition and physical composition. In abstract composition, services are selected according to their functionalities and context related constraints are omitted. In physical composition, concrete services are selected by considering context information. Yen et al. [15] proposed automated service composition reasoning techniques to handle dynamically arising situations. They consider that the specification of the current and desired states of the physical world is important to successful service composition reasoning. A model for the states of the physical world was also presented by them. Mohammed et al. [23] investigated the service composition problem under uncertainty in CPSs. They proposed the Markov task network, which can utilize the advantages of both hierarchical task network and Markov logic network to accomplish the service composition under uncertainty. Wang et al. [17] proposed a static service composition mechanism that exploits the workflow business logic model to build an abstract process graph of the task. In the situations where the task needs to be verified before it is executed, the service composition is often conducted in design time. In this composition approach, Petri net is often employed to model CPS services [9].

3. **Problem Definition.**

1) CPS service model: We let $\mathbf{S}$ denote a set of M CPS services in the system: $\mathbf{S} = \{\mathcal{S}_m | m = 1, 2, \ldots, M\}$. The m-th CPS service, $\mathcal{S}_m$, is associated with a 4-tuple: $< F_m, T_m, A_m, P_m >$, where F_m denotes the functional ability of $\mathcal{S}_m$, which includes functional category, e.g., removing obstacles, transporting objects, and functional properties, e.g., weight limit, and size limit, T_m denotes the operating time of $\mathcal{S}_m$, A_m denotes whether $\mathcal{S}_m$ needs a transportation service, and P_m denotes the PE that $\mathcal{S}_m$ needs. The set of all the transportation services in $\mathbf{S}$ is denoted as $\mathbf{S^R}$. $\mathbf{P}$ denotes the set of H PEs: $\mathbf{P} = \{\mathcal{P}_h | h = 1, 2, \ldots, H\}$, where $\mathcal{P}_h$ is the h-th PE.

2) Task model: We let $\mathbf{T}$ denote a set of N tasks in the system: $\mathbf{T} = \{\mathbf{T}_n | n = 1, 2, \ldots, N\}$. The n-th task, $\mathbf{T}_n$, consists of a sequence of K_n subtasks: $\mathbf{T}_n = \{\mathcal{T}_{n,k} | k = 1, 2, \ldots, K_n\}$. $\mathcal{T}_{n,k}$ means the k-th subtask of the n-th task, and is associated with a 4-tuple: $< F_{n,k}, B^l_{n,k}, E^l_{n,k}, \mathbf{S}_{n,k} >$. $F_{n,k}$ denotes the functional requirement of $\mathcal{T}_{n,k}$, which has the same definition as the functional ability of a CPS service. $B^l_{n,k}$ and $E^l_{n,k}$ denote the begin and end locations of the subtask $\mathcal{T}_{n,k}$, respectively. $\mathbf{S}_{n,k}$ denotes a set of CPS services that can accomplish $\mathcal{T}_{n,k}$. By comparing $F_{n,k}$ with the functional abilities of all the CPS services, $\mathbf{S}_{n,k}$ can be obtained before runtime.

3) Hypotheses: Hypotheses considered in this paper are summarized as follows: (1) the attributes of CPS services, e.g., functional properties, and operating time, are all known beforehand, and do not change at runtime; (2) a CPS service can only work for one task at a time; (3) when a CPS service is invoked to work for a task, it cannot be interrupted; (4) tasks and their locations are all known beforehand; (5) CPS services clearly know how to move their PEs between task locations, and the moving distances do not change at runtime; (6) a PE can only be used by one CPS service at a time; (7) the working area is relatively large, and PEs, such as robots, and drones, can autonomously avoid path conflicts between them.

4) Problem formulation: In this paper, we aim to find a service selection and invocation scheme that can optimize the makespan (the time for completing all tasks) as much as possible. We denote the makespan as Z. The problem can be formulated into the mixed integer programming as follows:

$$\min Z \tag{1}$$

$$\text{subject to: } Z \geq B^t_{n,k} + T_{n,k},\ n = 1, 2, \ldots, N,\ k = 1, 2, \ldots, K_n \tag{2}$$

$$\sum\nolimits_{1 \leq m \leq M} X_{m,n,k} * r(m,n,k) = 1,\ n = 1, 2, \ldots, N,\ k = 1, 2, \ldots, K_n \tag{3}$$

$$\sum\nolimits_{1 \leq m \leq M} Y_{m,n,k} * r(m,n,k) = A_{s(n,k)},\ n = 1, 2, \ldots, N,\ k = 1, 2, \ldots, K_n \tag{4}$$

$$\sum\nolimits_{1 \leq x \leq N, 1 \leq a \leq K_x, x \neq y \wedge a \neq b} \left(U^{s,s}_{P_{s(y,b)},x,a,y,b} + U^{r,s}_{P_{s(y,b)},x,a,y,b} \right) + W_{P_{s(y,b)},y,b} = X_{m,y,b},$$
$$y = 1, 2, \ldots, N,\ b = 1, 2, \ldots, K_y,\ m = 1, 2, \ldots, M \tag{5}$$

$$\sum\nolimits_{1 \leq x \leq N, 1 \leq a \leq K_x, x \neq y \wedge a \neq b} \left(U^{s,r}_{P_{sr(y,b)},x,a,y,b} + U^{r,r}_{P_{sr(y,b)},x,a,y,b} \right) + W^r_{P_{sr(y,b)},y,b} = Y_{m,y,b},$$
$$y = 1, 2, \ldots, N,\ b = 1, 2, \ldots, K_y,\ m = 1, 2, \ldots, M \tag{6}$$

$$\sum\nolimits_{1 \leq y \leq N, 1 \leq b \leq K_y, y \neq x \wedge b \neq a} \left(U^{s,s}_{P_{s(x,a)},x,a,y,b} + U^{s,r}_{P_{s(x,a)},x,a,y,b} \right) + V_{P_{s(x,a)},x,a} = X_{m,x,a},$$
$$x = 1, 2, \ldots, N,\ a = 1, 2, \ldots, K_x,\ m = 1, 2, \ldots, M \tag{7}$$

$$\sum\nolimits_{1 \leq y \leq N, 1 \leq b \leq K_y, y \neq x \wedge b \neq a} \left(U^{r,s}_{P_{s(x,a)},x,a,y,b} + U^{r,r}_{P_{s(x,a)},x,a,y,b} \right) + V^r_{P_{sr(x,a)},x,a} = Y_{m,x,a},$$
$$x = 1, 2, \ldots, N,\ a = 1, 2, \ldots, K_x,\ m = 1, 2, \ldots, M \tag{8}$$

$$B^t_{y,b} - B^t_{x,a} + D * \left(1 - U^{s,s}_{h,x,a,y,b}\right) >= T_{x,a} + \delta\left(E^l_{x,a}, B^l_{y,b}\right),$$
$$x, y = 1, 2, \ldots, N,\ a = 1, 2, \ldots, K_x,\ b = 1, 2, \ldots, K_y,\ h = 1, 2, \ldots, H \tag{9}$$

$$B^{t,r}_{y,b} - B^t_{x,a} + D * \left(1 - U^{s,r}_{h,x,a,y,b}\right) >= T_{x,a} + \delta\left(E^l_{x,a}, p(y,b)\right),$$
$$x, y = 1, 2, \ldots, N,\ a = 1, 2, \ldots, K_x,\ b = 1, 2, \ldots, K_y,\ h = 1, 2, \ldots, H \tag{10}$$

$$B^t_{y,b} - B^{t,r}_{x,a} + D * \left(1 - U^{r,s}_{h,x,a,y,b}\right) >= T^r_{x,a} + \delta\left(B^l_{x,a}, B^l_{y,b}\right),$$
$$x, y = 1, 2, \ldots, N,\ a = 1, 2, \ldots, K_x,\ b = 1, 2, \ldots, K_y,\ h = 1, 2, \ldots, H \tag{11}$$

$$B^{t,r}_{y,b} - B^{t,r}_{x,a} + D * \left(1 - U^{r,r}_{h,x,a,y,b}\right) >= T^r_{x,a} + \delta\left(B^l_{x,a}, p(y,b)\right),$$
$$x, y = 1, 2, \ldots, N,\ a = 1, 2, \ldots, K_x,\ b = 1, 2, \ldots, K_y,\ h = 1, 2, \ldots, H \tag{12}$$

$$B^t_{n,(k+1)} - B^t_{n,k} >= T_{n,k},\ n = 1, 2, \ldots, N,\ k = 1, 2, \ldots, K_n - 1 \tag{13}$$

$$B^t_{n,k} - B^{t,r}_{n,k} >= T^r_{n,k},\ n = 1, 2, \ldots, N,\ k = 1, 2, \ldots, K_n \tag{14}$$

$$\sum\nolimits_{1 \leq n \leq N, 1 \leq k \leq K_n} \left(V_{h,n,k} + V^r_{h,n,k}\right) <= 1,\ h = 1, 2, \ldots, H \tag{15}$$

$$\sum\nolimits_{1 \leq n \leq N, 1 \leq k \leq K_n} \left(W_{h,n,k} + W^r_{h,n,k}\right) <= 1,\ h = 1, 2, \ldots, H \tag{16}$$

$$B^{t,r}_{n,k} * (1 - A_{s(n,k)}) = 0,\ n = 1, 2, \ldots, N,\ k = 1, 2, \ldots, K_n \tag{17}$$

$$0 \leq B^t_{n,k} \leq +\infty,\ 0 \leq B^{t,r}_{n,k} \leq +\infty,\ n = 1, 2, \ldots, N,\ k = 1, 2, \ldots, K_n \tag{18}$$

$$X_{m,n,k} \in \{0,1\},\ Y_{m,n,k} \in \{0,1\},\ n = 1, 2, \ldots, N,\ k = 1, 2, \ldots, K_n,\ m = 1, 2, \ldots, M \tag{19}$$

$$U^{s,s}_{h,x,a,y,b} \in \{0,1\},\ U^{r,s}_{h,x,a,y,b} \in \{0,1\},\ U^{s,r}_{h,x,a,y,b} \in \{0,1\},\ U^{r,r}_{h,x,a,y,b} \in \{0,1\},$$
$$x, y = 1, 2, \ldots, N,\ a = 1, 2, \ldots, K_x,\ b = 1, 2, \ldots, K_y,\ h = 1, 2, \ldots, H \tag{20}$$

$$V_{h,n,k} \in \{0,1\},\ V^r_{h,n,k} \in \{0,1\},\ W_{h,n,k} \in \{0,1\},\ W^r_{h,n,k} \in \{0,1\},$$

$$n = 1, 2, \ldots, N,\ k = 1, 2, \ldots, K_n \tag{21}$$

The decision variables include $B^t_{n,k}$, $B^{t,r}_{n,k}$, $X_{m,n,k}$, $Y_{m,n,k}$, $U^{s,s}_{h,x,a,y,b}$, $U^{r,s}_{h,x,a,y,b}$, $U^{s,r}_{h,x,a,y,b}$, $U^{r,r}_{h,x,a,y,b}$, $V_{h,n,k}$, $V^r_{h,n,k}$, $W_{h,n,k}$ and $W^r_{h,n,k}$. We let $\mathcal{T}^r_{n,k}$ denote the transportation subtask that helps accomplish $\mathcal{T}_{n,k}$. $B^t_{n,k}$ and $B^{t,r}_{n,k}$ are the non-negative continuous variables that indicate the begin time of $\mathcal{T}_{n,k}$ and $\mathcal{T}^r_{n,k}$ respectively. The other ten are binary variables. $X_{m,n,k} = 1$ if CPS service $\mathcal{S}_m$ is selected to work for $\mathcal{T}_{n,k}$, and $Y_{m,n,k} = 1$ if transportation service $\mathcal{S}_m$ is selected to help accomplish $\mathcal{T}_{n,k}$. $U^{s,s}_{h,x,a,y,b}$, $U^{r,s}_{h,x,a,y,b}$, $U^{s,r}_{h,x,a,y,b}$ and $U^{r,r}_{h,x,a,y,b}$ define the order in which two selected CPS services use $\mathcal{P}_h$. For example, $U^r_{h,x,a,y,b} = 1$ if $\mathcal{P}_h$ is used by the transportation service selected to help accomplish $\mathcal{T}_{x,a}$ before it is used by the CPS service selected to accomplish $\mathcal{T}_{y,b}$, otherwise 0. $V_{h,n,k} = 1$ if the CPS service selected to work for $\mathcal{T}_{n,k}$ is the last one to use $\mathcal{P}_h$, otherwise 0. $V^r_{h,n,k} = 1$ if the transportation service selected to help accomplish $\mathcal{T}_{n,k}$ is the last one to use $\mathcal{P}_h$, otherwise 0. $W_{h,n,k} = 1$ if the CPS service selected to work for $\mathcal{T}_{n,k}$ is the first one to use $\mathcal{P}_h$, otherwise 0. $W^r_{h,n,k} = 1$ if the transportation service selected to help accomplish $\mathcal{T}_{n,k}$ is the first one to use $\mathcal{P}_h$, otherwise 0. It is worth noticing that the invocation sequence for each selected CPS service can be determined using subtask execution sequences, because subtasks are finally accomplished by CPS services.

The other symbols used are listed in Table 1. From constraint (2), we can see that the optimization objective, minimizing Z, is equivalent to making Z the upper bound for the end times of all subtasks. Constraint (3) ensures that for each subtask, exactly one CPS service is selected to work for it. Constraint (4) guarantees that if a selected CPS service needs a transportation service, exactly one transportation service is selected to

Table 1. Some symbols used in the paper

Symbol	Meaning
$B^l_{n,k}$	the begin location of $\mathcal{T}_{n,k}$
$E^l_{n,k}$	the end location of $\mathcal{T}_{n,k}$
$\mathcal{T}^r_{n,k}$	the transportation subtask that helps accomplish $\mathcal{T}_{n,k}$
$B^t_{n,k}$	the begin time of $\mathcal{T}_{n,k}$
$B^{t,r}_{n,k}$	the begin time of $\mathcal{T}^r_{n,k}$
$E^t_{n,k}$	the end time of $\mathcal{T}_{n,k}$. $E^t_{n,k} = B^t_{n,k} + T_{n,k}$
$r(m,n,k)$	$r(m,n,k) = 1$ if $\mathcal{S}_m$ belongs to $\mathbf{S}_{n,k}$ (the candidate service set of $\mathcal{T}_{n,k}$), otherwise 0
$T_{n,k}$	the operating time of the CPS service that is selected to accomplish $\mathcal{T}_{n,k}$. $T_{n,k} = \sum_{1\le m\le M} X_{m,n,k} * T_m * r(m,n,k)$
$T^r_{n,k}$	the operating time of the CPS service that is selected to accomplish $\mathcal{T}^r_{n,k}$. $T^r_{n,k} = \sum_{1\le m\le M} Y_{m,n,k} * T_m * r(m,n,k)$
$s(n,k)$	the CPS service selected to work for $\mathcal{T}_{n,k}$. $s(n,k) = \sum_{1\le m\le M} X_{m,n,k} * m$
$sr(n,k)$	the transportation selected to help accomplish $\mathcal{T}_{n,k}$. $sr(n,k) = \sum_{1\le m\le M} Y_{m,n,k} * m$
$p(n,k)$	the end location of the subtask that uses $\mathcal{P}_{P_{s(n,k)}}$ before $\mathcal{T}_{n,k}$. $p(n,k) = \sum_{1\le x\le N, 1\le a\le K_x, x\ne n\wedge a\ne k} \left(U^{s,s}_{P_{s(n,k)},x,a,n,k} * E^l_{x,a}\right)$
$\delta(l_1,l_2)$	the moving time from the location l_1 to the location l_2
$[p,q]$	the time interval that begins at p and ends at q

help it; if otherwise, no transportation service is selected. Constraints (5)-(8) ensure that all selected CPS services that are not the first or last one to use PEs have predecessors or successors respectively. Constraints (9)-(12) enforce that there is enough time for CPS services to accomplish subtasks and move (or transport) PEs between task locations. Constraint (13) guarantees that for each task, its subtasks are accomplished in sequence. Constraint (14) enforces that before a selected CPS service begins to be executed, the required PE must be transported to the task location. In constraints (9)-(12) D is a large positive value that helps formulate mixed integer programming constraints.

It is easily seen that solving the complicated mixed integer programming is a nontrivial task. Therefore, we turn to heuristic approaches in the sequel.

4. **Proposed Approach Based on the Improved Quantum Genetic Algorithm.** Quantum genetic algorithm (QGA) [24, 25] has been proved to be an effective algorithm for the complicated optimization problems such as task scheduling [26], and job shop scheduling [27, 28]. Compared to classic evolutionary algorithms, QGA can maintain better population diversity with smaller population size by introducing the basic ideas behind quantum computing, such as quantum bits, and superposition states. Thus, Q-GA can achieve superior results in a relatively low computation cost. Inspired by these superiorities of QGA, we propose to utilize QGA to design a heuristic approach.

The proposed heuristic approach is based on the improved quantum genetic algorithm for service selection and invocation (called SSI-IQGA). The improvements of SSI-IQGA to the basic QGA include dynamic rotation angle mechanism, quantum variation and quantum crossover.

The workflow of SSI-IQGA is depicted in Figure 2. The input data of SSI-IQGA includes CPS services, tasks, environmental information (e.g., task locations and moving

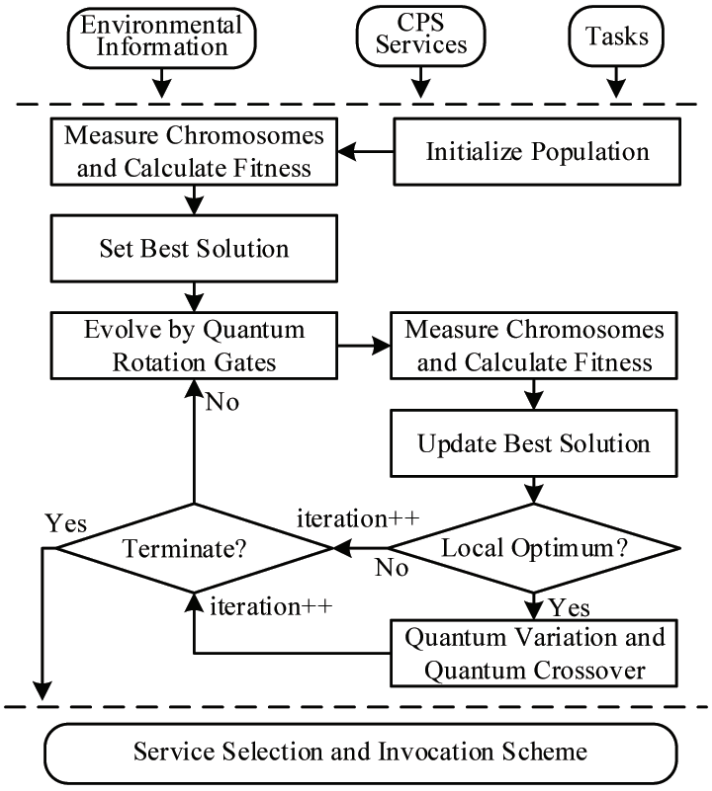

FIGURE 2. The workflow of SSI-IQGA

paths between them). The population of SSI-IQGA is initialized at the beginning of the algorithm. In each evolution iteration, the population is first evolved by quantum rotation gates. Afterwards, every chromosome in the population is measured and a set of feasible solutions is obtained. Then, the quality of each feasible solution is evaluated by calculating its fitness. Next, the best solution is updated based on the fitness values. At last, to increase the probability of SSI-IQGA to escape from local optimality, quantum variation and quantum crossover is conducted in some evolution iterations. The algorithm terminates when the maximum evolution iteration number is achieved. The output is the service selection and invocation scheme obtained from the best solution of SSI-IQGA.

In the subsequent subsections, we first introduce the basic QGA, and then detail the main steps of SSI-IQGA. At last, we discuss the implementation of SSI-IQGA.

4.1. **Introduction to QGA.** Quantum bit (qubit) and quantum superposition state are the basic concepts of QGA. A qubit is represented by a superposition of two basis states $|0\rangle$ and $|1\rangle$: $|q\rangle = \alpha|0\rangle + \beta|1\rangle$, $\alpha^2 + \beta^2 = 1$, where α and β are complex numbers, denoting the probability amplitudes of the basis states.

The fundamental difference between QGA and the traditional genetic algorithm (GA) is that in QGA each chromosome in the population is made up of multiple qubits. For example, a chromosome ψ containing L qubits can be described by

$$\psi = \begin{array}{c} \begin{array}{cccc} q_1 & q_2 & \cdots & q_L \end{array} \\ \left[\begin{array}{c|c|c|c} \alpha_1 & \alpha_2 & \ldots & \alpha_L \\ \beta_1 & \beta_2 & \ldots & \beta_L \end{array}\right] \end{array}, \tag{22}$$

where α_j and β_j are the probability amplitudes of qubit q_j $(j = 1, 2, \ldots, L)$. In general, after the measurement of a chromosome, each qubit of the chromosome will collapse into a certain state ($|0\rangle$ or $|1\rangle$), and a feasible solution will be obtained finally. However, we cannot determine which certain state a qubit will collapse into, so a chromosome in QGA can correspond to different feasible solutions. It makes QGA have better population diversity than GA.

4.2. **Solution representation.** In our SSI-IQGA algorithm, the qubits of a chromosome fall into three parts: CPS service selection part, transportation service selection part and subtask sequence part. The qubits in the first part indicate the CPS services that are selected to work for the subtasks. Since one qubit can only collapse into one of two certain states ($|0\rangle$ or $|1\rangle$), if a subtask has more than two candidate services, more than one qubit are required to represent the selected CPS service. In general, if a subtask has n candidate services, the number of qubits required can be calculated as $\lceil \log_2(n) \rceil$, where $\lceil x \rceil = \min\{z | z \geq x, z \in \mathbb{Z}^+\}$. The qubits in the second part indicate the transportation services that are selected to help the selected CPS services. It has the same representation manner as the first part. In the second part, if a selected CPS service does not need a transportation service, one qubit is still required, but the state of the qubit is meaningless in the decoding procedure.

The qubits in the last part indicate the execution sequence of the subtasks from different tasks. Because only binary numbers can be directly represented in QGA, a converting mechanism is required to represent the subtask sequence. Assume that the total number of subtasks is D, and each subtask is assigned with a one-dimensional index ranging from 0 to $D-1$. The total number of qubits required in the last part is $\lceil \log_2(D) \rceil * D$, where $\lceil x \rceil = \min\{z | z \geq x, z \in \mathbb{Z}^+\}$, and each $\lceil \log_2(D) \rceil$ qubit corresponds to a subtask. After the measurement of the chromosome, the qubits in the last part will form a binary string of length $\lceil \log_2(D) \rceil * D$. The converting mechanism works as follows. First, each $\lceil \log_2(D) \rceil$

bit of the binary string is converted into a decimal number. In this way, a decimal string of length D is obtained, and every decimal number in it also corresponds to a subtask. Afterwards, the subtasks are sorted in the ascending order according to these decimal numbers. If two subtasks have the same decimal number, we let the subtask with smaller index in front of the other one. At last, for each task, its subtasks exchange their positions with each other to ensure that they are accomplished in sequence.

For the example in Figure 1, Table 2 lists the tasks and CPS services, and a possible solution to it is shown in Figure 3. Because in the example problem each subtask has two candidate services at most, each qubit in the first (second) part corresponds to a selected CPS service (transportation service). The third qubits in the first and second parts collapse into $|0\rangle$ and $|1\rangle$ respectively. It means that $\mathcal{S}_1$ (the 0-th candidate service of $\mathcal{T}_{2,1}$) and $\mathcal{S}_4$ (the 1st candidate transportation service) are selected to accomplish $\mathcal{T}_{2,1}$. Each three qubits in the last part correspond to a subtask. The subtask sequence can be determined by first sorting the subtasks in the ascending order according to the decimal numbers converted from the formed binary string. After that, for each task the positions of its subtasks are exchanged to ensure that these subtasks are accomplished in sequence.

TABLE 2. The tasks and CPS services in the motivation example. $\mathcal{S}_3$ and $\mathcal{S}_4$ are transportation services, and they are also the candidate services of both $\mathcal{T}_{1,2}$ and $\mathcal{T}_{2,3}$. $\mathcal{T}_{1,1}$, $\mathcal{T}_{2,1}$ and $\mathcal{T}_{2,2}$ all have only one candidate service. Each subtask is assigned with a one-dimensional index ranging from 0 to 4.

Tasks	**Subtasks**	**Candidate services**			
		$\mathcal{S}_1$	$\mathcal{S}_2$	$\mathcal{S}_3$	$\mathcal{S}_4$
$\mathbf{T}_1$(Task 1)	$\mathcal{T}_{1,1}$(RO)(0)		√		
	$\mathcal{T}_{1,2}$(TO)(1)			√	√
$\mathbf{T}_2$(Task 2)	$\mathcal{T}_{2,1}$(PF)(2)	√			
	$\mathcal{T}_{2,2}$(RO)(3)		√		
	$\mathcal{T}_{2,3}$(TO)(4)			√	√

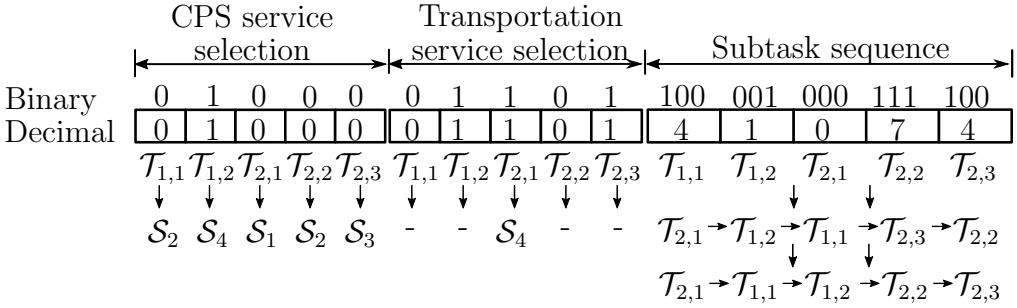

FIGURE 3. A possible solution to the example problem

4.3. **Initialization of SSI-IQGA.** We denote the population size as G and the number of qubits required by each chromosome as L. Denote the population after the t-th evolution iteration as $\Psi^{[t]} = \left\{\psi_1^{[t]}, \psi_2^{[t]}, \ldots, \psi_G^{[t]}\right\}$, where the i-th chromosome $\psi_i^{[t]}$ $(i = 1, 2, \ldots, G)$ is as described in (22). At the beginning of SSI-IQGA, every chromosome in the population is initialized by making the two states of each qubit appear with the same probability. That is, we set

$$\alpha_{i,j}^{[0]} = \beta_{i,j}^{[0]} = 1/\sqrt{2},\ i = 1, 2, \ldots, G,\ j = 1, 2, \ldots, L, \tag{23}$$

and then obtain

$$\psi_i^{[0]} = \begin{bmatrix} \alpha_{i,1}^{[0]} & \alpha_{i,2}^{[0]} & \dots & \alpha_{i,L}^{[0]} \\ \beta_{i,1}^{[0]} & \beta_{i,2}^{[0]} & \dots & \beta_{i,L}^{[0]} \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & \dots & 1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} & \dots & 1/\sqrt{2} \end{bmatrix}, \ i = 1, 2, \dots, G. \tag{24}$$

4.4. **Measurement.** Since a qubit is represented by a superposition of two basic states, the certain state of a qubit cannot be determined until we measure it. The measurement of a chromosome is to make each qubit of the chromosome collapse into a certain state so that a feasible solution can be obtained. The measurement can be done in two steps. In the first step, we generate a random number ranging from 0 to 1 for each qubit of the chromosome, and then we compare the random number with $|\alpha|^2$. If the random number is less than $|\alpha|^2$, the qubit collapses into $|0\rangle$, otherwise $|1\rangle$. In this way, a binary string of L bits is formed at the end of the first step.

In the second step, we check whether the binary string is exactly a feasible solution and repair it if not. What causes the generation of infeasible binary strings is that decimal numbers cannot be directly used to represent the service selection and invocation scheme, and in the situations where decimal numbers must be used, a decimal number can only be expressed by a group of qubits. For a subtask with five candidate services, three qubits are required to express the selected service, assuming that each candidate service is assigned with an index ranging from 0 to 4. However, after each of these three qubits collapses into a certain state, we cannot ensure that the value of the formed binary string must be less than or equal to 4. If the value is greater than 4, an infeasible binary string is generated. In this study, an infeasible binary string is repaired by replacing each out-of-range value with a random but reasonable value.

We denote by Υ_i the feasible solution obtained after the measurement of the i-th chromosome: $\Upsilon_i = \{\upsilon_{i,1}, \upsilon_{i,2}, \dots, \upsilon_{i,L}\}$. According to the solution representation manner ahead, a service selection and invocation scheme can be obtained from each feasible solution.

4.5. **Fitness calculation.** The quality of a feasible solution is usually called the fitness. For the LSMC problem, the fitness is closely related to the makespan, since the final objective is to optimize the makespan. Owing to it, the fitness of a feasible solution is calculated as $1.0/Z$ in SSI-IQGA, assuming that the makespan of the feasible solution is denoted by Z. Then, what becomes important is how to calculate the makespan based on a known service selection and invocation scheme.

To facilitate us to calculate the makespan, a task sequence graph (TSG) is utilized. In a TSG, each PE is associated with an ordered subtask set (see Figure 4). Based on the subtask sets, the service selection conflicts can be resolved, because in essence, the service selection conficts are caused by the contentions of PEs between CPS services. The makespan calculation is accomplished by inserting all the subtasks into the subtask sets of the TSG one by one according to the subtask sequence. There are two basic types of subtask insertion procedures in total. The newly introduced symbols are listed in Table 1.

1) Type 1: For a subtask $\mathcal{T}_{n,k}$, if the selected CPS service $\mathcal{S}_m$ does not need a transportation service, inserting $\mathcal{T}_{n,k}$ is the simplest. We need to check all the idle time intervals of $\mathcal{P}_h$ (the PE required by $\mathcal{S}_m$) according to time order. If an idle time interval, $[E_{y,a}^t, B_{x,c}^t]$, is the earliest one that can satisfy

$$B_{x,c}^t - \eta >= T_{n,k} + \delta\left(E_{n,k}^l, B_{x,c}^l\right), \tag{25}$$

where $\eta = \max\left\{E_{n,(k-1)}^t, E_{y,a}^t + \delta\left(E_{y,a}^l, B_{n,k}^l\right)\right\}$, $\mathcal{T}_{n,k}$ will be inserted into this time interval (see Figure 4). η indicates the begin time of $\mathcal{T}_{n,k}$, and ensures that $\mathcal{T}_{n,k}$ can only begin

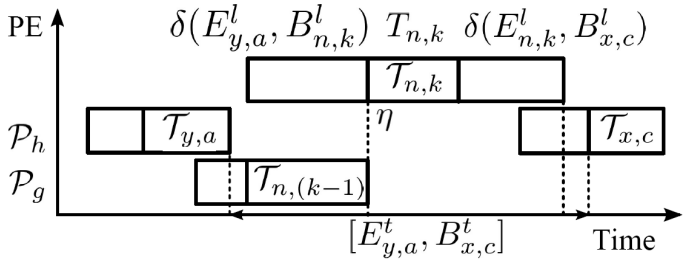

FIGURE 4. A subtask insertion procedure of the first type

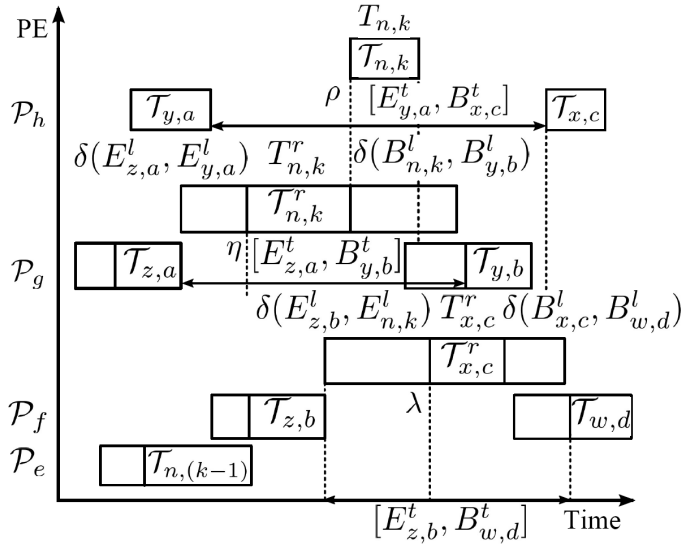

FIGURE 5. A subtask insertion procedure of the second type

to be executed after $\mathcal{T}_{n,(k-1)}$ ends. After $\mathcal{T}_{n,k}$ ends, $\mathcal{P}_h$ must have enough time to move to $B^l_{x,c}$ before $\mathcal{T}_{x,c}$ begins. This constraint is ensured by (25).

2) Type 2: If a transportation service $\mathcal{S}_t$ is selected to help $\mathcal{S}_m$, and $\mathcal{S}_m$ just needs a transportation service, inserting $\mathcal{T}_{n,k}$ is much more complex. We denote the required PEs of $\mathcal{S}_m$ and $\mathcal{S}_t$ as $\mathcal{P}_h$ and $\mathcal{P}_g$, respectively. In this situation, we not only need to find an appropriate idle time interval for $\mathcal{T}_{n,k}$, but also need to find appropriate idle time intervals for the transportation subtasks of $\mathcal{T}_{n,k}$ and $\mathcal{T}_{x,c}$ (see Figure 5). Therefore, inserting $\mathcal{T}_{n,k}$ can be accomplished by two steps. In the first step, we check all the idle time intervals of $\mathcal{P}_h$ and $\mathcal{P}_g$ according to time order. If an idle time interval set, $< \left[E^t_{y,a}, B^t_{x,c}\right], \left[E^t_{z,a}, B^t_{y,b}\right] >$, can satisfy

$$B^t_{y,b} - \eta \geq T^r_{n,k} + \delta\left(B^l_{n,k}, B^l_{y,b}\right), \tag{26}$$

$$B^t_{x,c} - \rho \geq T_{n,k}, \tag{27}$$

we proceed to the second step. If not, we continue to check the next idle time interval set. In (26) and (27), $\eta = \max\left\{E^t_{y,a}, E^t_{z,a} + \delta\left(E^l_{z,a}, E^l_{y,a}\right)\right\}$, $\rho = \max\left\{E^t_{n,(k-1)}, \eta + T^r_{n,k}\right\}$. η indicates the begin time of $\mathcal{T}^r_{n,k}$ (the transportation subtask that helps to complete $\mathcal{T}_{n,k}$), and ensures that $\mathcal{T}^r_{n,k}$ can only begin to be executed after $\mathcal{T}_{y,a}$ ends. ρ indicates the begin time of $\mathcal{T}_{n,k}$, and ensures that $\mathcal{T}_{n,k}$ can only begin to be executed after both $\mathcal{T}^r_{n,k}$ and $\mathcal{T}_{n,(k-1)}$ end.

We denote the transportation service that is used to transport $\mathcal{P}_h$ for $\mathcal{T}_{x,c}$ as $\mathcal{S}_{t'}$, and the PE used by $\mathcal{S}_{t'}$ as $\mathcal{P}_f$. In the second step, we need to check whether there is an idle time interval of $\mathcal{P}_f$ during which $\mathcal{S}_{t'}$ can transport $\mathcal{P}_h$ to $B^l_{x,c}$ before $\mathcal{T}_{x,c}$ begins. To achieve it, we can first remove the existing transportation subtask of $\mathcal{T}_{x,c}$ from the subtask set of $\mathcal{P}_f$, and then check all the idle time intervals of $\mathcal{P}_h$ and $\mathcal{P}_f$ according to time order. If an idle time interval set, $< \left[E^t_{y,a}, B^t_{x,c}\right], \left[E^t_{z,b}, B^t_{w,d}\right] >$, can satisfy

$$B^t_{w,d} - \lambda \geq T^r_{x,c} + \delta\left(B^l_{x,c}, B^l_{w,d}\right), \tag{28}$$

$$B^t_{x,c} - \lambda \geq T^r_{x,c}, \tag{29}$$

it means that $\mathcal{T}^r_{x,c}$ (the new transportation subtask of $\mathcal{T}_{x,c}$) can be inserted into $[E^t_{z,b}, B^t_{w,d}]$. At this point, the idle time intervals that are fit to $\mathcal{T}_{n,k}$ and the related subtasks are found. In (28) and (29), $\lambda = \max\left\{\rho + T_{n,k}, E^t_{z,b} + \delta\left(E^l_{z,b}, E^l_{n,k}\right)\right\}$, it indicates the begin time of $\mathcal{T}^r_{x,c}$, and ensures that the begin time of $\mathcal{T}^r_{x,c}$ must be later than the end time of $\mathcal{T}_{n,k}$. (29) ensures that $\mathcal{T}^r_{x,c}$ must end before $\mathcal{T}_{x,c}$ begins to be executed.

It is worth noticing that in illustrating these insertion procedures, we only consider the cases where the idle time intervals are between two successive subtasks, and for type 2, we only consider the cases where $\mathcal{P}_g$ and $\mathcal{P}_f$ are different PEs. As for other cases, we only need to make several minor modifications to these procedures.

4.6. **Evolution by quantum rotation gates.** The population can be evolved by quantum rotation gates. Based on the quantum rotation gate $\Omega(\theta_{i,j})$, the j-th qubit of the i-th chromosome ψ_i can be updated as follows:

$$\begin{bmatrix} \alpha'_{i,j} \\ \beta'_{i,j} \end{bmatrix} = \Omega(\theta_{i,j}) \begin{bmatrix} \alpha_{i,j} \\ \beta_{i,j} \end{bmatrix} = \begin{bmatrix} \cos(\theta_{i,j}) & -\sin(\theta_{i,j}) \\ \sin(\theta_{i,j}) & \cos(\theta_{i,j}) \end{bmatrix} \begin{bmatrix} \alpha_{i,j} \\ \beta_{i,j} \end{bmatrix}, \tag{30}$$

where $\alpha'_{i,j}$ and $\beta'_{i,j}$ denote the probability amplitudes of the j-th qubit after update. The rotation angle $\theta_{i,j}$ is denoted as

$$\theta_{i,j} = s(\alpha_{i,j}, \beta_{i,j}) * \Delta\theta_{i,j}, \tag{31}$$

where $s(\alpha_{i,j}, \beta_{i,j})$ determines the quantum rotation direction for the j-th qubit, and $\Delta\theta_{i,j}$ determines the magnitude of the rotation angle for the j-th qubit. In our SSI-IQGA algorithm, to reduce the impact of the quantum rotation on the algorithm convergence rate, the rotation angle is dynamically adjusted as the population evolves. The specific adjustment policies are listed in Table 3, where σ is a coefficient closely related to the evolution iteration number. Specially, σ is denoted as

$$\sigma = 0.04\pi\left(1 - 0.5 * \frac{t}{t^{\max} + 1}\right), \tag{32}$$

where t is the current evolution iteration number and $t^{\max}$ is the maximum number of evolution iterations.

4.7. **Quantum variation and quantum crossover.** Quantum variation and quantum crossover can produce new generation populations, thus increasing the probability of SSI-IQGA to escape from local optimality. In this study, quantum variation works as follows. First, we select a small number of chromosomes from the population with the variation probability p^v. Afterwards, for each chromosome that is selected, we pick up a random variation position from all the qubits, and then exchange the probability amplitudes α and β in the variation position.

Quantum crossover produces new child chromosomes by swapping the qubits of parent chromosomes. By inheriting good qubits, child chromosomes tend to be better than their parents. In this study, we adopt the two-point crossover [29], which is achieved

TABLE 3. Adjustment policies for the rotation angle. Υ^b denotes the current best solution, and υ_j^b is the j-th bit of Υ^b. Υ_i denotes the i-th feasible solution, and $\upsilon_{i,j}$ is the j-th bit of Υ_i. $F(\cdot)$ means the fitness value.

$\upsilon_{i,j}$	υ_j^b	$F(\Upsilon_i) \geq F(\Upsilon^b)$	$\Delta\theta_{i,j}$	$s(\alpha_{i,j}, \beta_{i,j})$			
				$\alpha_{i,j}\beta_{i,j} > 0$	$\alpha_{i,j}\beta_{i,j} < 0$	$\alpha_{i,j} = 0$	$\beta_{i,j} = 0$
0	0	False	0	0	0	0	0
0	0	True	0	0	0	0	0
0	1	False	0	0	0	0	0
0	1	True	σ	-1	1	± 1	0
1	0	False	σ	-1	1	± 1	0
1	0	True	σ	1	-1	0	± 1
1	1	False	σ	1	-1	0	± 1
1	1	True	σ	1	-1	0	± 1

Algorithm 1: SSI-IQGA

Input: Task set, CPS service set, Environmental information, Population size G, Variation probability p^v, Crossover probability p^c, Maximum evolution iteration number $t^{\max}$

Output: Service selection and invocation scheme

```
Set the evolution iteration number t = 0;
Generate the initial population Ψ^[0] = {ψ_1^[0], ψ_2^[0], ..., ψ_G^[0]};
Measure every chromosome in the population Ψ^[t] and obtain a set of solutions
  O^[t] = {Υ_1, Υ_2, ..., Υ_G};
foreach Υ_i ∈ O^[t] do
    Calculate the fitness F(Υ_i);
Set Υ^b = arg max_{Υ∈O^[t]} F(Υ);
while t <= t^max do
    Set t = t + 1;
    Evolve the population by quantum rotation gates;
    Perfrom steps 3 to 5;
    Set Υ^{b,c} = arg max_{Υ∈O^[t]} F(Υ);
    if F(Υ^b) < F(Υ^{b,c}) then
        Υ^b = Υ^{b,c};
    if Υ^b does not change during ⌊t^max/10⌋ evolution iterations then
        Do quantum crossover with the probability p^c for Ψ^[t];
        Do quantum variation with the probability p^v for Ψ^[t];
Obtain the service selection and invocation scheme from Υ^b;
```

by first generating two random crossover points. Afterwards, the parent chromosomes exchange the qubits between the crossover points with each other, and then two children chromosomes are produced.

4.8. **Implementation of SSI-IQGA.** Although the main steps of SSI-IQGA are detailed above, there are still some points worthy of notice. Firstly, in each evolution iteration, after all the fitness values are calculated, we need to find the best solution in

all the solutions just obtained and check whether the best solution is better than the old one. If it is true, the old best solution needs to be replaced with the new one. Secondly, if the best solution does not change during $\lfloor t^{\max}/10 \rfloor$ evolution iterations, we consider that the population is very likely to be trapped into local optimality. Therefore, quantum crossover and quantum variation need to be conducted to produce new generation populations. Thirdly, SSI-IQGA terminates when the maximum evolution iteration number is achieved. The maximum evolution iteration number is determined by experiments, which is detailed in Section 5.2. The pseudo code of SSI-IQGA is presented in Algorithm 1.

Fourthly, one advantage of QGA is that it can be easily transformed into a parallel algorithm that can effectively take the advantage of current multi-core CPUs to reduce the computation time. The basic idea behind the parallelization of the algorithm is splitting the (global) population into a set of local populations and allowing each local population evolve independently. In this study, parallel versions of our SSI-IQGA algorithm are also implemented. The workflow of the parallel SSI-IQGA algorithm (SSI-IQGA-P) is shown in Figure 6. In each evolution iteration two synchronization operations need to be conducted. The first one is to make all local populations share the same best solution. The last one is to migrate chromosomes between different local populations. Although only two threads are drawn in Figure 6, the parallel SSI-IQGA algorithm can utilize any number of threads to speed up the computation.

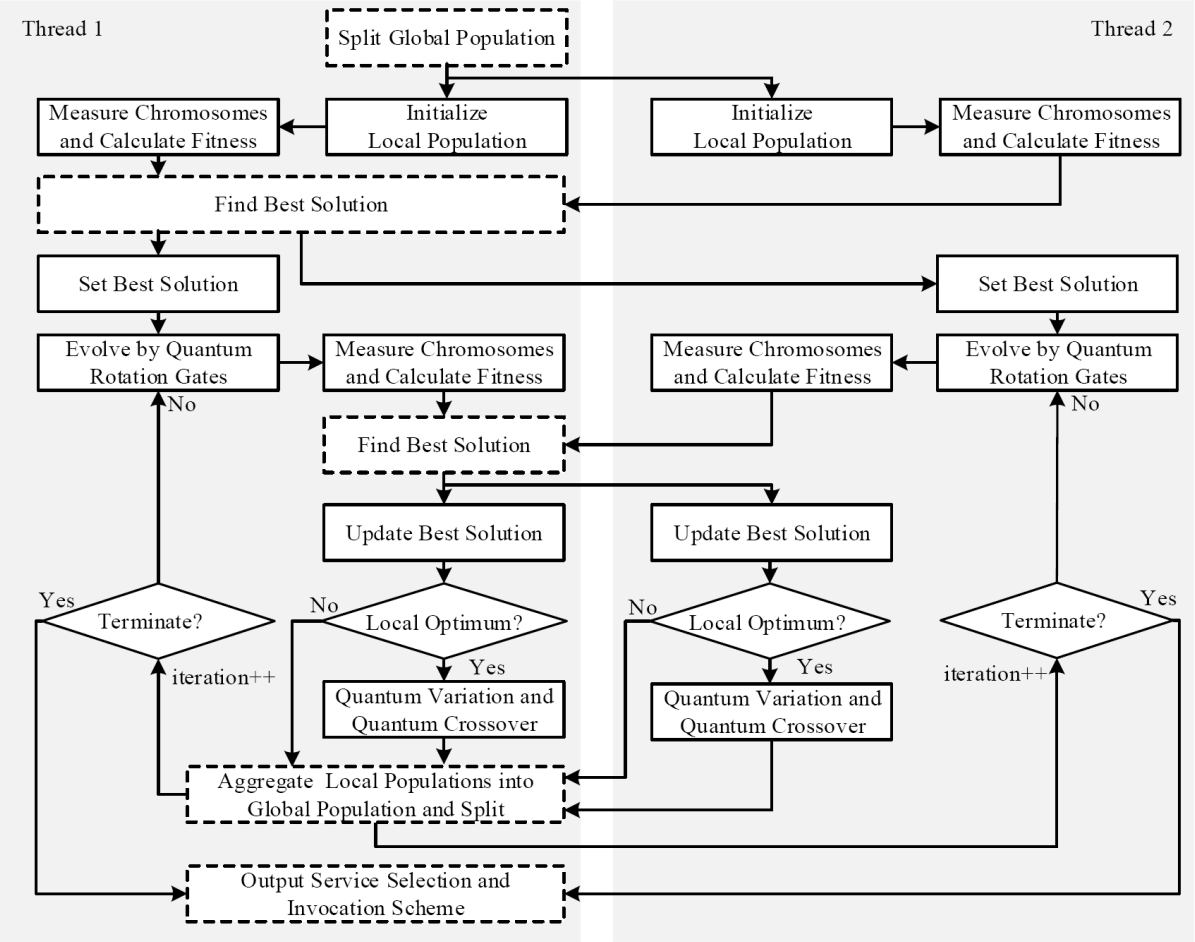

FIGURE 6. The workflow of the parallel SSI-IQGA algorithm. The rectangles with dashed borders represent the synchronization operation.

5. Performance Evaluation.

5.1. Experiment settings and baselines.

1) Experiment Settings: To evaluate the performance of the proposed algorithm, we carry out simulations on smart emergency response scenarios. Similar to the motivation example in Section 1, a number of valuable objects need to be transferred to a safe place under each of these scenarios. In the simulations, the tasks and their subtasks are generated based on the Gehring-Homberger benchmark [30]. The benchmark consists of 300 problem instances in total, and a set of physical locations is defined for every problem instance. These physical locations are used as the begin and end locations of the subtasks. A task has 2 to 3 subtasks. For some subtasks, their begin and end locations are the same, while for the others, their begin and end locations are different. The physical location number of a problem instance varies from 200 to 1000 based on the scale of the working area. These sets of physical locations have three types of spatial distribution in total: clustered distribution, scattered distribution, and hybrid distribution.

The CPS services in the simulations are randomly generated. Only a part of the CPS services need a transportation service. If a CPS service is not a transportation service, its operating time is set as $d * e$, where d indicates the service time defined in the benchmark, and e is randomly selected from the set $\{0.5, 0.8, 1, 1.2, 1.5\}$. In terms of functional ability, there are three types of CPS services in total. The candidate services of a subtask are randomly selected from all the CPS services according to its functional requirement, and a subtask has four candidate services at most.

The moving time between two locations is defined as the Euclidean distance between them. The operating time of a transportation service is defined as the moving time plus a quarter of the service time defined in the benchmark.

2) Baselines: Two algorithms are used as baselines in our experiments. The first one is the basic quantum genetic algorithm (QGA). The second one is the greedy algorithm (GR). The proposed algorithm and baselines are implemented in C++ language.

5.2. Evaluation of makespan.

1) Impact of algorithm parameters: There are four parameters to be adjusted to improve the performance of SSI-IQGA: (1) the maximum evolution iteration number ($t^{\max}$); (2) the population size (G); (3) the variation probability (p^v); and (4) the crossover probability (p^c). To evaluate the impact of each parameter, we generate four groups of parameters configuration as in Table 4. For each configuration, we tune one parameter and fix the other parameters, and the simulations are conducted 100 times in order to obtain the average value.

Table 4. Parameters of configuration

Configuration	$t^{\max}$	G	p^v	p^c
Config1	20-180	50	0.1	0.8
Config2	100	10-90	0.1	0.8
Config3	100	50	0.02-0.5	0.8
Config4	100	50	0.1	0.5-0.9

Figure 7 shows the results of tuning different algorithm parameters. We can find that the performance of SSI-IQGA increases as the maximum evolution iteration number (or population size) goes up. Meanwhile, when the maximum iteration number (or population size) exceeds a certain value (e.g., $t^{\max} = 100$, $G = 50$ in Figure 7), the performance improvement of SSI-IQGA is not significant. However, it is not the same case for the

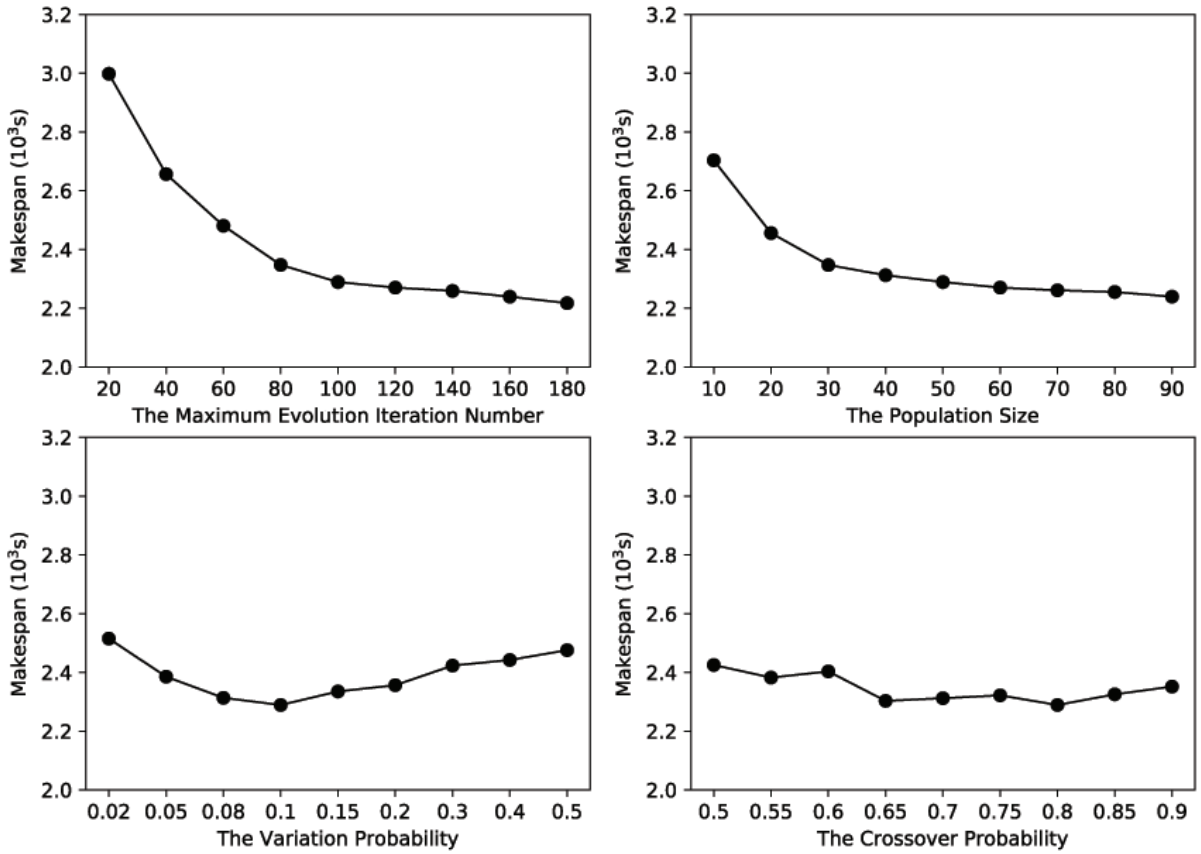

FIGURE 7. Impact of different algorithm parameters

variation probability. When the variation probability exceeds a certain value (e.g., $p^v = 0.1$ in Figure 7), the makespan achieved begins to rise. It indicates that too many quantum variations are not good for the performance improvement of SSI-IQGA. The last observation is that the crossover probability does not impact the performance of SSI-IQGA significantly as the other three parameters do. However, in the whole, raising the crossover probability is still useful to slightly improve the performance of SSI-IQGA.

According to these findings, the maximum evolution iteration number, the population size, the variation probability and the crossover probability are set as 100, 50, 0.1 and 0.8 respectively in the following evaluations.

2) Different types of spatial distribution: In Figure 8 we compare the performance of the proposed algorithm and baselines in terms of makespan using 30 problem instances with three types of spatial distribution. Each problem instance used is constructed using a problem instance in the Gehring-Homberger benchmark. For each problem instance, we conduct 100 times of experiments to get the average value. We can see that SSI-IQGA outperforms QGA and GR under all types of spatial distribution. The parallel SSI-IQGA algorithm (SSI-IQGA-P) achieves the similar performance of SSI-IQGA in terms of makespan.

3) Different number of CPS services and tasks: We also evaluate the performance of SSI-IQGA, QGA and GR under different number of CPS services and tasks. We show the results in Figures 9 and 10 respectively. We use three instances in the comparisons: c101, r105, rc104, and the experiments are also conducted 100 times in order to obtain the average value. From the results, we can see that the makespan of SSI-IQGA is still lower than that of QGA and GR under different situations.

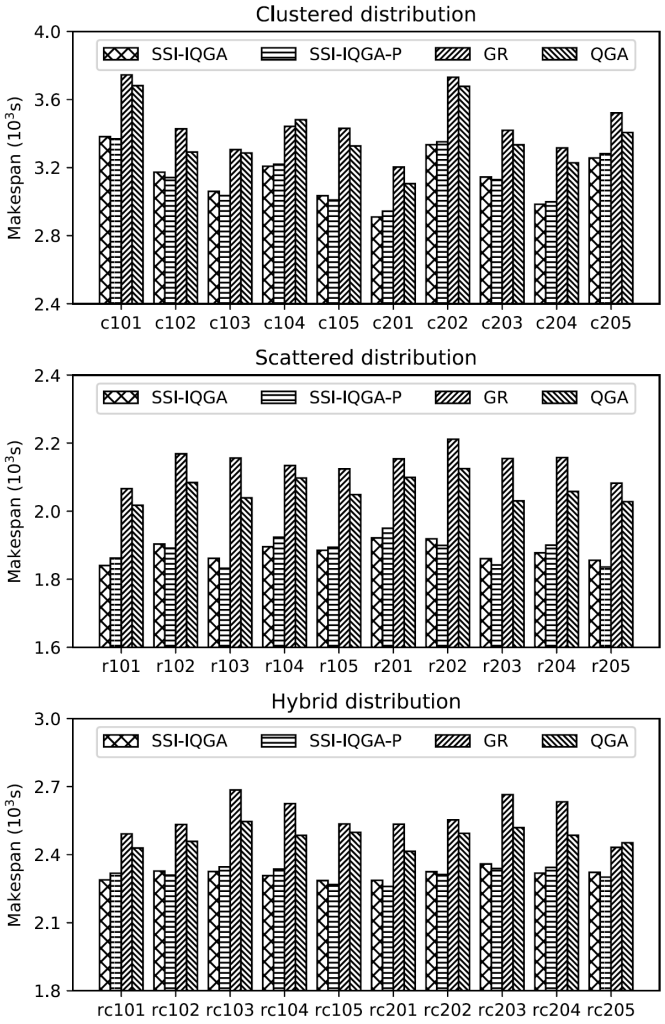

FIGURE 8. Makespan under different types of spatial distribution. The CPS service number M is 40, and the task number N is 200. The physical location number of each instance is 200.

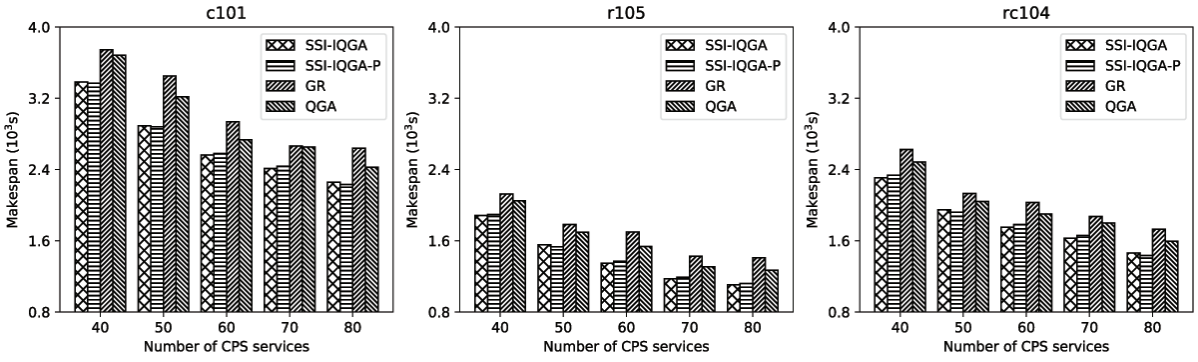

FIGURE 9. Makespan under different number of CPS services. The task number N is 200. The physical location number for each instance is 200.

For a given number of tasks, with the rising of the CPS service number, there will be more candidate services for tasks. Owing to it, for each task, its completion time is also very likely to decrease because its probability of contending with other tasks for CPS services will become smaller. Therefore, with the increasing number of tasks that can be

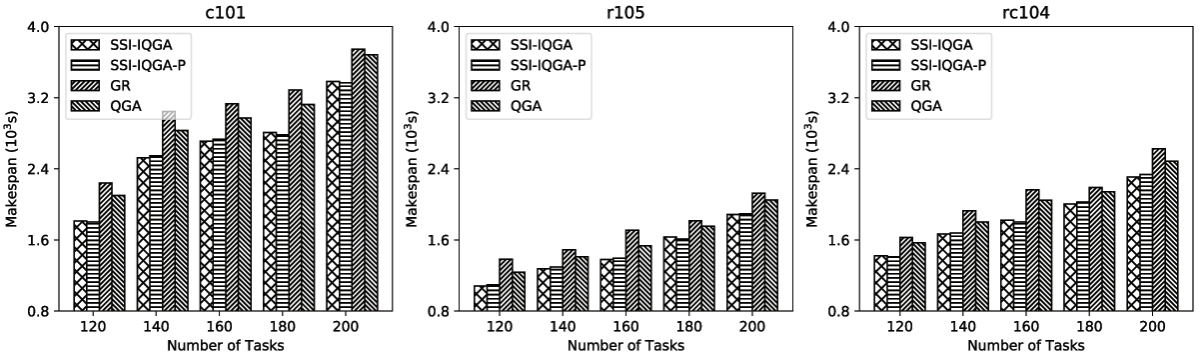

FIGURE 10. Makespan under different number of tasks. The CPS service number M is 40. The physical location number for each instance is 200.

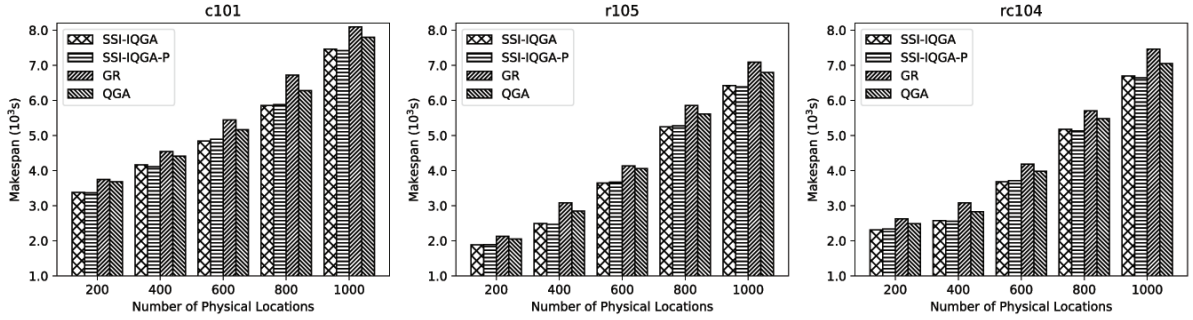

FIGURE 11. Makespan under different number of physical locations. The CPS service number M is 40, and the task number N is 200.

completed earlier, there is a huge possibility that the makespan will reduce. The finding can be seen from Figure 9. On the contrary, for a given number of CPS services, the probability of contending for CPS services between tasks increases with the number of tasks rising. As a sequence, the makespan is most likely to go up (see Figure 10).

4) Different number of physical locations: Afterwards, we evaluate the impacts of the physical location number on the performance of the proposed algorithm and baselines. For each problem instance, we keep unchanged all the attributes of each CPS service and each task under different number of physical locations except the task locations and the operating time of transportation services. We run each of these algorithms 100 times to get the average value. We show the results in Figure 11. We can find that SSI-IQGA can also find better service selection and invocation schemes than baselines under different situations.

In general, the more physical locations there are, the larger the working area is. It also means the moving time between two locations is very likely to increase. Therefore, there is a huge possibility that the makespan will go up. We can also see this finding from Figure 11.

5) Summary: SSI-IQGA outperforms QGA and GR in terms of effectiveness. The superiority of SSI-IQGA to QGA also demonstrates that the search performance of SSI-IQGA is enhanced by such improvements as dynamic rotation angle mechanism, quantum variation and quantum crossover. Specifically, with dynamic rotation angle mechanism, SSI-IQGA can adjust the rotation angle to enhance population diversity and reduce the influence of the rotation angle on its convergence rate; with quantum variation and quantum crossover, SSI-IQGA can produce new generation populations when necessary, thus increasing the probability of SSI-IQGA to escape from local optimality.

5.3. **Evaluation of consumed time.** The consumed time of the proposed algorithm and baselines is shown in Figure 12. The experiments are conducted on a workstation with Intel Xeon(R) 2.0GHz 8 processors, 64 GB RAM and 64-bit Ubuntu operating system. We run each of these algorithms 100 times to get the average value. As shown, with the increasing number of CPS services and tasks, the time cost of both QGA and SSI-IQGA raises more slowly than GR. It can be seen that QGA has lower computation time than SSI-IQGA. The reason for it is that the improvements of SSI-IQGA to QGA (dynamic rotation angle mechanism, quantum variation and quantum crossover) incur extra time consumption.

We also find that the parallel SSI-IQGA algorithm (SSI-IQGA-P) can effectively utilize the potential of current multi-core CPUs to save the computation time (see Figure 12). In SSI-IQGA-P, local populations evolve independently. In general, the computation workload for evolving each local population of SSI-IQGA-P decreases with the number of local populations rising. Therefore, the overall time cost also decreases as the number of local population increases. This finding can be seen from Figure 13, which shows the speedup ratio achieved under different number of local populations. Another observation from Figure 13 is that the speedup ratio increases almost linearly as the number of local population goes up.

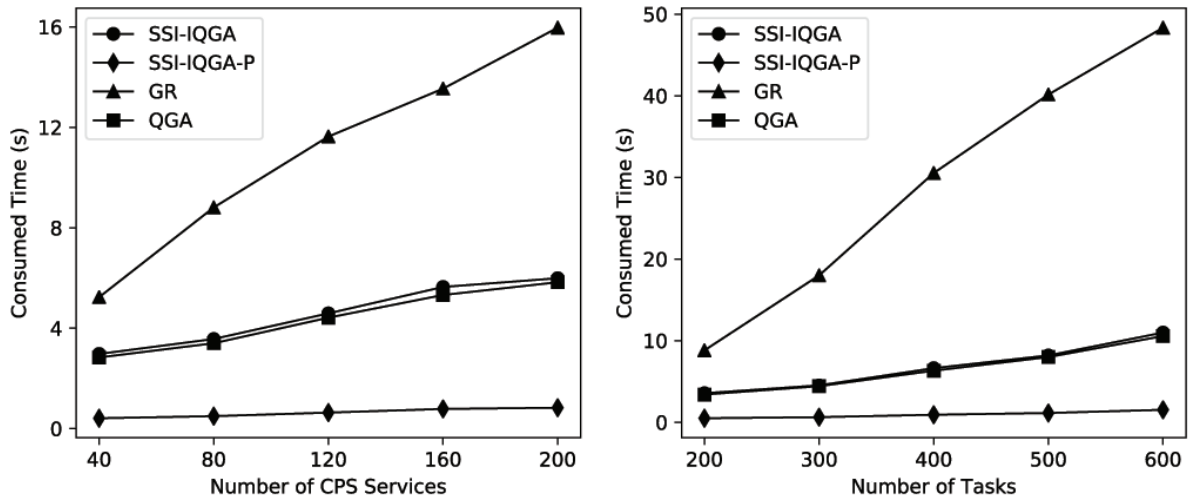

FIGURE 12. Comparison of consumed time of different algorithms. The number of local populations in SSI-IQGA-P is eight. Left – The task number N is 200. Right – The CPS service number M is 80.

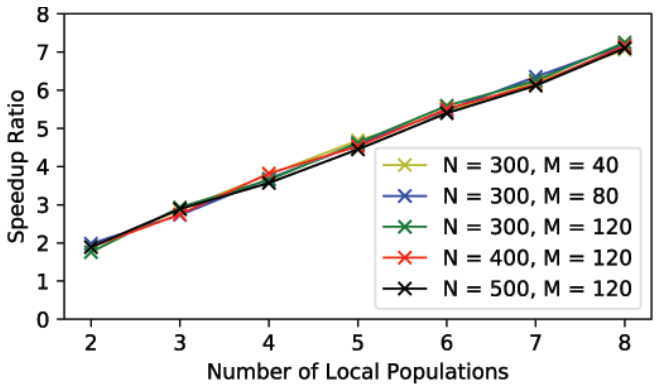

FIGURE 13. Speedup achieved under different number of local populations

In summary, the advantage of the proposed algorithm is clear, and it can be used to efficiently find superior service selection and invocation schemes.

6. **Conclusion.** In this paper we investigate the location sensitive multi-task oriented CPS service composition problem. Since the core mission of the problem is to find a service selection and invocation scheme than can optimize the makespan as much as possible, a heuristic approach based on the improved quantum genetic algorithm is presented to address it. The simulation results show that the proposed method outperforms other methods under different situations, and can be used to efficiently obtain superior service selection and invocation schemes. In the future, we will consider other characteristics of CPSs that might influence existing CPS service composition approaches, such as more general transportation services.

REFERENCES

[1] E. A. Lee, Cyber physical systems: Design challenges, *Proc. of the 11th IEEE Int. Symp. on Obj. & Compo.-Orient. Real-Time Distrib. Comput.*, pp.363-369, 2008.

[2] L. Sha, S. Gopalakrishnan, X. Liu and Q. Wang, Cyber-physical systems: A new frontier, *Proc. of IEEE Int. Conf. on Sens. Netw. Ubiquit. & Trus. Comput.*, pp.1-9, 2008.

[3] R. R. Rajkumar, I. Lee, L. Sha and J. Stankovic, Cyber-physical systems: The next computing revolution, *Proc. of the 47th Des. Automat. Conf.*, pp.731-736, 2010.

[4] I. L. Yen, G. Zhou, W. Zhu, F. Bastani and S. Y. Hwang, A smart physical world based on service technologies, big data, and game-based crowd sourcing, *Proc. of IEEE Int. Conf. on Web Serv.*, pp.765-772, 2015.

[5] S. K. Khaitan and J. D. McCalley, Design techniques and applications of cyberphysical systems: A survey, *IEEE Syst. J.*, vol.9, no.2, pp.350-365, 2015.

[6] K. Kamei, S. Nishio, N. Hagita and M. Sato, Cloud networked robotics, *IEEE Network*, vol.26, no.3, pp.28-34, 2012.

[7] A. Taherkordi and F. Eliassen, Towards independent in-cloud evolution of cyber-physical systems, *Proc. of IEEE Int. Conf. on Cyb.-Phys. Syst. Netw. & Appl.*, pp.19-24, 2014.

[8] B. Kehoe, S. Patil, P. Abbeel and K. Goldberg, A survey of research on cloud robotics and automation, *IEEE Trans. Autom. Sci. Eng.*, vol.12, no.2, pp.398-409, 2015.

[9] J. M. Mendes, P. LeitAco, F. Restivo and A. W. Colombo, Composition of petri nets models in service-oriented industrial automation, *Proc. of the 8th IEEE Int. Conf. on Ind. Inform.*, pp.578-583, 2010.

[10] D. Guinard, V. Trifa, S. Karnouskos, P. Spiess and D. Savio, Interacting with the SOA-based Internet of things: Discovery, query, selection, and on-demand provisioning of web services, *IEEE Trans. Serv. Comput.*, vol.3, no.3, pp.223-235, 2010.

[11] R. Seiger, C. Keller, F. Niebling and T. Schlegel, Modelling complex and flexible processes for smart cyber-physical environments, *J. Comput. Sci.*, vol.10, pp.137-148, 2015.

[12] W. Zhu, G. Zhou, I. L. Yen and F. Bastani, A PT-SOA model for CPS/IoT services, *Proc. of IEEE Int. Conf. on Web Serv.*, pp.647-654, 2015.

[13] Y. Cai, Z. Tang, Y. Ding and B. Qian, Theory and application of multi-robot service-oriented architecture, *IEEE/CAA J. Autom. Sin.*, vol.3, no.1, pp.15-25, 2016.

[14] J. Huang, F. Bastani, I. L. Yen and J. J. Jeng, Toward a smart cyber physical space: A context-sensitive resource-explicit service model, *Proc. of the 33rd Annu. IEEE Int. Comput. Softw. & Appl. Conf.*, pp.122-127, 2009.

[15] I. L. Yen, W. Zhu, F. Bastani, Y. Huang and G. Zhou, Rapid service composition reasoning for agile cyber physical systems, *Proc. of IEEE Symp. on Serv.-Orient. Syst. Eng.*, pp.442-449, 2016.

[16] J. Huang, F. B. Bastani, I. L. Yen and W. Zhang, A framework for efficient service composition in cyber-physical systems, *Proc. of IEEE Symp. on Serv.-Orient. Syst. Eng.*, pp.291-298, 2010.

[17] T. Wang, L. Cheng and K. Zhang, Automatic and effective service provision with context-aware service composition mechanism in cyber-physical systems, *Adv. Inform. Sci. & Serv. Sci.*, vol.4, no.11, pp.151-160, 2012.

[18] D. A. Menasce, QoS issues in web services, *IEEE Internet Comput.*, vol.6, no.6, pp.72-75, 2002.

[19] J. Huang, F. Bastani, I. L. Yen, J. Dong, W. Zhang, F. J. Wang and H. J. Hsu, Extending service model to build an effective service composition framework for cyber-physical systems, *Proc. of IEEE the 7th Int. Conf. on Serv.-Orient. Comput. & Appl.*, pp.1-8, 2009.

[20] D. Martin, M. Paolucci and S. McIlraith, Bringing semantics to web services: The OWL-S approach, *Proc. of Int. Workshop on Seman. Web Serv. & Web Proc. Compos.*, pp.26-42, 2005.

[21] X. Jin, S. Chun, J. Jung and K.H. Lee, IoT service selection based on physical service model and absolute dominance relationship, *Proc. of IEEE the 7th Int. Conf. on Serv.-Orient. Comput. & Appl.*, pp.65-72, 2014.

[22] K. Wan, V. Alagar and Y. Dong, Specifying resource-centric services in cyber physical systems, *Proc. of Int. MultiConf. of Eng. & Comput. Sci.*, pp.83-97, 2014.

[23] A. W. Mohammed, Y. Xu, H. Hu and B. Agyemang, Markov task network: A framework for service composition under uncertainty in cyber-physical systems, *Sensors*, vol.16, no.9, p.1542, 2016.

[24] K. H. Han and J. H. Kim, Quantum-inspired evolutionary algorithm for a class of combinatorial optimization, *IEEE Trans. Evol. Comput.*, vol.6, no.6, pp.580-593, 2002.

[25] A. Malossini, E. Blanzieri and T. Calarco, Quantum genetic optimization, *IEEE Trans. Evol. Comput.*, vol.12, no.2, pp.231-241, 2008.

[26] D. Konar, S. Bhattacharyya, K. Sharma, S. Sharma and S. R. Pradhan, An improved hybrid quantum-inspired genetic algorithm (HQIGA) for scheduling of real-time task in multiprocessor system, *Appl. Soft Comput.*, vol.53, pp.296-307, 2017.

[27] J. Gu, X. Gu and M. Gu, A novel parallel quantum genetic algorithm for stochastic job shop scheduling, *J. Math. Anal. Appl.*, vol.355, no.1, pp.63-81, 2009.

[28] T. Ning, H. Jin, X. Song and B. Li, An improved quantum genetic algorithm based on MAGTD for dynamic FJSP, *J. Amb. Intel. Hum. Comp.*, pp.1-10, 2017.

[29] M. Watanabe, K. Ida and M. Gen, A genetic algorithm with modified crossover operator and search area adaptation for the job-shop scheduling problem, *Comput. Ind. Eng.*, vol.48, no.4, pp.743-752, 2005.

[30] D. S. Vianna, L. S. Ochi and L. M. A. Drummond, A parallel hybrid evolutionary metaheuristic for the period vehicle routing problem, *Proc. of the 10th Symp. on Paral. & Distrib. Proces.*, pp.183-191, 1999.