Subtask Scheduling for Distributed Robots in Cloud Manufacturing

Wenxiang Li, *Member, IEEE*, Chunsheng Zhu, *Student Member, IEEE*, Laurence T. Yang, *Member, IEEE*, Lei Shu, *Member, IEEE*, Edith C.-H. Ngai, *Member, IEEE*, and Yajie Ma, *Member, IEEE*

***Abstract*—Due to the limitation of capacity in an enterprise, cooperation among these enterprises is necessary to handle a complex production task. Cloud manufacturing (CMF) provides a cooperation platform for efficient utilization of distributed manufacturing resources in regional enterprise cluster. However, effective scheduling of tasks or subtasks to these resources is a challenging problem. Based on the analysis on the procedure of task processing, this paper proposes a CMF scheduling model for efficiently exploiting distributed resources, so industrial robots in different enterprises can cooperatively handle a batch of tasks. Specifically, this paper considers the performance of four robot deployment methods, including random deployment, robot-balanced deployment, function-balanced deployment, and location-aware deployment. Furthermore, three subtask-scheduling strategies are derived for three optimization objectives, including load-balance of robots, minimizing overall cost, and minimizing overall processing time. Moreover, these strategies are implemented by genetic algorithm. Simulation results demonstrate that each strategy can achieve the relevant optimization objective. In addition, the results also show that the physical distance between two enterprises can influence the overall cost, and location-aware deployment leads to smaller transportation cost. Location-aware deployment and function-balanced deployment lead to smaller overall processing time for the low-workload state and high-workload state of the system, respectively.**

***Index Terms*—Cloud manufacturing (CMF), deployment, genetic algorithm (GA), manufacturing robot, task scheduling.**

I. Introduction

ROBOT-REPRESENTED manufacturing resources signify expensive investment and operational cost, and it is infeasible for extensive enterprises to deploy these resources [1]. Nowadays, cooperative manufacturing among regional enterprise clusters in urban agglomeration develops rapidly. It can avoid out-of-order expansion of production capacity and has become a new trend of current manufacturing industry. Although there are convenient logistics services among these enterprises, insufficiency of resource-sharing and cooperation platforms leads to inefficient utilization of manufacturing resources and capacities, and it is necessary to exploit these resources effectively.

Cloud manufacturing (CMF) [2] combines current manufacturing platforms with cloud computing techniques and provides virtual functions (such as description, registration, and distribution) for exploiting dispersed manufacturing resources. CMF can enhance production capacity, provide efficient cooperation among production chain and enterprise clusters, and foster the conversion from production-oriented to service-oriented manufacturing industry. With CMF, production capacity becomes an on-demand resource for various enterprises to search, match, and utilize [3].

It is necessary to apply intelligent configurations on CMF to control the progress or states of real-time production. However, tremendous user-scale, varying user-demands, and heterogeneity of manufacturing resources pose challenges to efficient resource utilization, and current task-scheduling methods cannot meet the demands of intelligent allocation and sharing of resources [4]. This paper explores the measures for these problems, and the main contributions are as follows.

1) Based on the analysis on the elements and procedure of production with cooperative industrial robots of different locations and functions, this paper proposes a CMF scheduling model that assigns a robot for each subtask rather than the whole task. This model can employ these dispersed robots for simultaneous processing a batch of tasks. Detailed descriptions on the features and relations of relevant entities are given in this model.

Manuscript received December 14, 2014; revised March 19, 2015; accepted May 5, 2015. Date of publication June 11, 2015; date of current version June 26, 2017. This work was supported by the National Natural Science Foundation of China under Grant 61104215, the Scientific Research Foundation for the Returned Overseas Chinese Scholars of State Education Ministry, the Science and Technology Research Project of the Education Department of Hubei Province, China under Grant D20151106 and Grant Q20141110, open fund from the Engineering Research Center of Metallurgical Automation and Detecting Technology of Ministry of Education, Wuhan University of Science and Technology (MARC201304), and the Natural Sciences and Engineering Research Council of Canada. The work of L. Shu was supported by 2013 Special Fund of Guangdong Higher School Talent Recruitment, Educational Commission of Guangdong Province, China, Project No. 2013KJCX0131, Guangdong High-Tech Development Fund 2013B010401035, 2013 top Level Talents Project in "Sailing Plan" of Guangdong Province, National Natural Science Foundation of China (Grant 61401107), and 2014 Guangdong Province Outstanding Young Professor Project. The work of E. C.-H. Ngai was supported in part by the SSF ProFuN project and Vinnova GreenIoT project in Sweden. *(Corresponding author: Chunsheng Zhu.)*

W. Li and Y. Ma are with the School of Information Science and Engineering, Engineering Research Center of Metallurgical Automation and Detecting Technology of Ministry of Education, Wuhan University of Science and Technology, Wuhan 430081, China (e-mail: liwx2006@hotmail.com; mayajie@wust.edu.cn).

C. Zhu is with the Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, BC V6T 1Z4, Canada (e-mail: cszhu@ece.ubc.ca).

L. T. Yang is with the Department of Computer Science, St. Francis Xavier University, Antigonish, NS B2G 2W5, Canada (e-mail: ltyang@stfx.ca).

L. Shu is with the Guangdong Provincial Key Laboratory of Petrochemical Equipment Fault Diagnosis, Guangdong University of Petrochemical Technology, Maoming 525000, China (e-mail: lei.shu@lab.gdupt.edu.cn).

E. C.-H. Ngai is with the Department of Information Technology, Uppsala University, 751 05 Uppsala, Sweden (e-mail: edith.ngai@it.uu.se).

Digital Object Identifier 10.1109/JSYST.2015.2438054

TABLE I
MAIN NOTATION DEFINITIONS

Notation	Definition
N_W	The number of warehouses
WH_t	The t-th warehouse
N_M	The number of manufacturing centers
MC_i	The i-th manufacturing centre
TC_i	Cost for MC_i to transport component or product
$D(u,v)$	Distance between location u and location v
N_R	The number of types of robots
$R_{i,j}$	Robot that belongs to the j-th type and resides in MC_i
$RN_{i,j}$	The number of $R_{i,j}$
$PD_{i,j}$	Time to handle an ERC* by $R_{i,j}$
$PC_{i,j}$	Cost to handle an ERC by $R_{i,j}$
N_{ST}	The number of types of subtasks
ST_n	Subtask of the n-th type
N_P	The number of types of processes
P_m	Process of the m-th type
STL_n	Average layer position of ST_n in all relevant processes
$TV_{m,n}$	Weight of product or component from ST_n in P_m
$PV_{m,n}$	The number of ERCs that the component of ST_n in P_m amounts to
$F[n][j]$	Function mapping between robot type j and subtask type n
N_T	The total number of tasks to be scheduled
T_k	The k-th task
Prc_k	The process type for T_k
$Prn_{k,n}$	The previous subtask in the process tree for ST_n of T_k
TM_k	The number of product items for T_k
$FP_{m,n}$	A Boolean variable indicating whether ST_n exists in P_m

* ERC: Equivalent Reference Component

2) With considerations for component or product transportation during manufacturing, this paper proposes several subtask-scheduling strategies for optimization objectives such as load-balance of robots, minimizing overall cost, and minimizing overall processing time. These strategies are implemented by genetic algorithm (GA), and simulation experiments have verified the effectiveness of these strategies.
3) This paper proposes several methods for deploying robots at different enterprise or manufacturing centers (MC for short), including random deployment, robot-balanced deployment, function-balanced deployment, and location-aware deployment. Simulation experiments have made comparison on the performance of these methods.

The main notation definitions are illustrated in Table I. The rest of this paper is organized as follows. Section II summarizes current works on scheduling in CMF. Section III describes the CMF system model and robot-deployment methods. Section IV proposes several subtask-scheduling strategies and relevant implementation methods. Section V illustrates the settings of the simulation experiment and analyzes the simulation results. At last, Section VI gives the conclusion.

II. RELATED WORKS

In recent years, many works have been done on scheduling in CMF. Reference [5] designed an intelligent scheduling system based on quality-predication method and proposed the procedure for scheduling operation. The scheduling engine inside it contained a sequential evaluation system for products, a deployment system, and a monitoring system. Reference [6] designed a cloud-center architecture of robots to provide services for everyone and proposed resource-scheduling algorithm for heterogeneous robots to meet users' demands with the minimal cost. Reference [7] proposed energy-adaptive immune GA for cooperative task-scheduling in CMF. The algorithm cannot only improve the diversity of searching results based on various immune tactics but also adjust the crossing and mutation probability self-adaptively for a good balance between diversification and intensification of searching results. Reference [8] took computation and communication costs into consideration, gave description of the time for dynamic resource allocation in CMF, and implemented the allocation method with GA. Reference [9] proposed a Qos-based selection model of manufacturing services and designed solutions based on cloud model and ant-colony optimization (ACO) method. Reference [10] dealt with negotiation-based task allocation of resources for dynamic scheduling in agent-based holonic control framework. All of the aforementioned works allocate the entire task to a distributed manufacturing unit. However, as the complexity of production grows, task decomposition and cooperation among manufacturing units for this task should be considered.

Reference [11] proved that an optimal production solution can be computed in polynomial time. Several polynomial time heuristics were designed for the most realistic specialized settings. Simulation results assessed their efficiency, showing that the best heuristics obtain a good production throughput. Reference [12] proposed on-demand task allocation strategy for workflow-based CMF, and by exploring the factors that affect resource allocation, such as cost, capacity, credibility, and load, the matching-method between these factors and task-demands were proposed. Considering the influence from material and information flows, [13] decomposed each task into many subtasks, built a resource-configuration model for minimizing cost and delay together with optimal product quality, and got solutions for this model based on max inherit optimization. Reference [14] proposed a task allocation method that matches demands and capacity. By allocating subtasks to proper resource providers, this method got a multiresolution hierarchical architecture of resources and capacity. Taking time and resources constraints into consideration, [15] implemented a detailed procedure of decomposition and scheduling of multiple tasks for cost minimization. Reference [16] took three phases for scheduling, i.e., narrowing problem-solving space by constraints such as delay, cost, and quality, implementing cloud service for each subtask by point-clustering algorithm, and adopting immune GA on the basis of artificial neural network to gain the optimal combination of partners. However, all of aforementioned works ignore the influences from product transportation among distributed manufacturing centers and deployment strategy of robots.

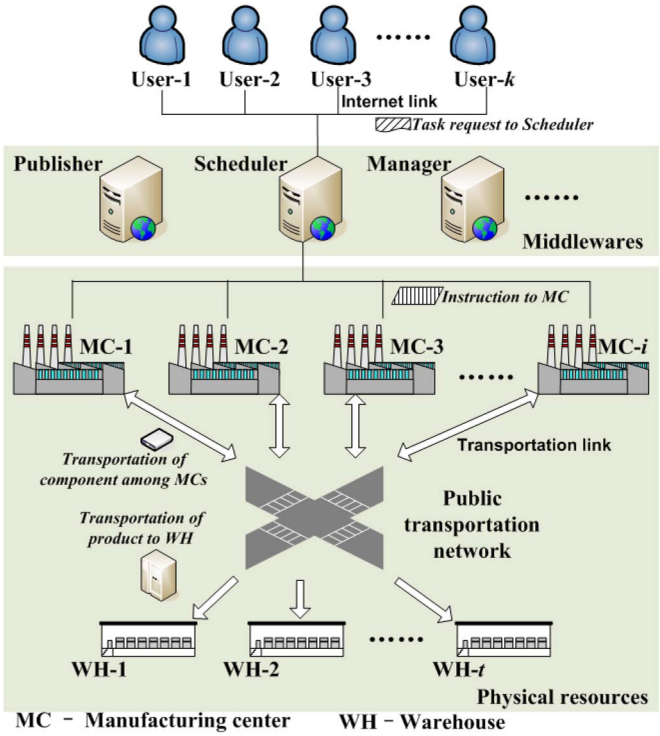


Fig. 1. Model of cooperative manufacturing systems.

III. Subtask Scheduling in CMF

A. Architecture of CMF Systems

In this paper, our work covers exactly from the beginning to the end of product manufacturing, so we neither consider the logistics of raw materials for robots nor consider that of final products for users. Considering the dispersed locations of manufacturing centers and distributed handling of tasks, we propose a cooperative manufacturing system model in Fig. 1. The processing procedure in this system is as follows. CMF is implemented by middlewares such as resource publisher, resource manager, and resource scheduler. Task requests from all users are sent to the sole robots scheduler via the Internet, and the scheduler begins to work periodically or on-demand. After analyzing the process of each task and the current states of the robots in the system, the scheduler allocates a robot for each subtask with instructions via the Internet. The robots in MCs begin to handle all subtasks that belong to them respectively. During the process of manufacturing, there are bidirectional transportation flows of components among MCs and unidirectional transportation flows of final products to prespecified warehouses (WH for short).

B. Description of CMF Scheduling Entities

1) Robot scheduler $\mathbf{S}$ provides three functions: 1) receiving task requests from the Internet; 2) analyzing and decomposing each task into many subtasks according to the process and allocating a robot to each subtask; and 3) sending the scheduling instructions to relevant MCs.
2) Warehouse $\mathbf{WH}_t$ ($0 \leq t < N_W$, and N_W is the total number of warehouses) stores the final products of the tasks.

	$R_{*,0}$	$R_{*,1}$	$R_{*,2}$	$R_{*,3}$
ST_0	1	0	0	0
ST_1	0	1	0	0
ST_2	0	1	1	0
ST_3	0	0	1	0
ST_4	0	0	0	1

Fig. 2. Function mapping between subtask type and robot type.

3) Manufacturing center $\mathbf{MC}_i$ ($0 \leq i < N_M$, and N_M is the total number of MCs) is equipped with some types of robots and has enough transportation capacity and raw material supplies. Its properties include the following: 1) unit transportation time $\mathbf{TD}_i$ represents the speed of the transportation facilities in MC_i for transporting components or products. 2) Unit transportation cost $\mathbf{TC}_i$ represents the cost of the transportation facilities in MC_i for transporting components or products. 3) $\mathbf{D}(\mathbf{u}, \mathbf{v})$ stands for the physical distance between location u and location v.
4) Given $0 \leq j < N_R$ (N_R is the total number of robot types), robot $\mathbf{R}_{i,j}$ stands for the robot that is of the jth type and resides in MC_i. Considering the heterogeneity of subtasks, we introduce a unified metric called equivalent reference component (ERC for short) as reference for measuring the relative cost and time for handling different subtasks, and we select a small component and assign the cost and time for handling it to ERC. The properties of $R_{i,j}$ include the following: 1) unit production time $\mathbf{PD}_{i,j}$ stands for the time needed by $R_{i,j}$ for producing one ERC. 2) Unit production cost $\mathbf{PC}_{i,j}$ stands for the cost needed by $R_{i,j}$ for producing one ERC. As a fixed performance parameter, the $\mathrm{PD}_{i,j}$'s are all identical for the robots of the same type. However, due to the difference of locations, the $\mathrm{PC}_{i,j}$ may be different among the robots of the same type in different MCs. 3) The number of robots of $R_{i,j}$ is $\mathbf{RN}_{i,j}$. 4) Measured by the number of ERCs, $\mathbf{CL}_{i,j}$ is the current workload left to be handled in $R_{i,j}$.
5) Given $0 \leq n < N_{\mathrm{ST}}$ (N_{ST} is the total number of subtask types), subtask of type n (labeled as $\mathbf{ST}_n$) is the minimal element to be handled in robots, and there is a function matrix (labeled as $\mathbf{F}[N_{\mathrm{ST}}][N_R]$) between ST_n and robot type $R_{*,j}$. As shown in Fig. 2, if a robot type can handle a subtask type, the corresponding matrix element is 1, and 0 otherwise.
6) Given $0 \leq m < N_P$ (N_P is the total number of process types), process of type m (labeled as $\mathbf{P}_m$) is the relation structure of relevant subtasks and the procedure for a complete product. There are complex relations among the subtasks in a process, so many structures can be used to describe the process type, including the fork structure. However, due to the assembling of components, these structures can be transformed into tree structure, which holds only the aggregation relationship and is easier to describe. For example, the components from subtask A can be used for two successive subtasks, i.e., B and C. This is a fork structure. We can divide A into two subtasks of the same type, i.e., A1 and A2. Moreover, the components from A1 and A2 are used for B and C, respectively. In

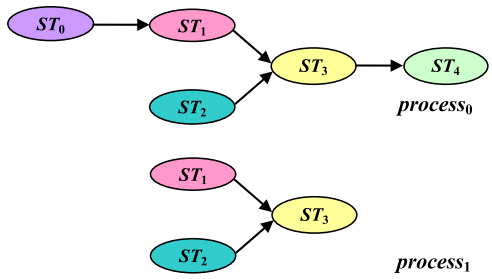


Fig. 3. Tree structure formation of processes.

this way we convert the fork structure into tree structure. As a tree node, each subtask is handled according to process specification, and the subtask that finishes last in the process is the root node of the tree. Fig. 3 is an example of tree structure formation for two different processes. The detailed properties are as follows.

1) Subtask's unit production output $\mathbf{PV}_{m,n}$ stands for the number of ERCs that the component of ST_n in P_m amounts to.
2) Subtask's unit transportation volume $\mathbf{TV}_{m,n}$ stands for the weight of components of ST_n in P_m.
3) Assign a Boolean variable $\mathrm{FP}_{m,n} = 1$ if ST_n exists in P_m and $\mathrm{FP}_{m,n} = 0$ otherwise, and let $L_{m,n}$ be the number of layers from ST_n to the root of process tree in P_m. Taking $\mathrm{process}_0$ in Fig. 3 as an example, ST_4 is at layer 0, and ST_1 is at layer 2. Therefore, the average layer position for ST_n in all relevant processes is

$$\mathrm{STL}_n = \frac{\sum_{m=0}^{N_P-1} \mathrm{FP}_{m,n} L_{m,n}}{\sum_{m=0}^{N_P-1} \mathrm{FP}_{m,n}}. \quad (1)$$

Obviously, the ST_n with larger STL_n tends to be handled earlier in the process.

7) Given $0 \le k < N_T$ (N_T is the number of tasks to be handled), the kth task of label $\mathbf{T}_k$ has the following properties: 1) the process type for T_k (labeled as $\mathbf{Prc}_k$); 2) the type of previous subtask for ST_n in Prc_k (labeled as $\mathbf{Prn}_{k,n}$); 3) the total number of product items (labeled as $\mathbf{TM}_k$); 4) the MC that ST_n of T_k is scheduled to (labeled as $\mathbf{Pos}_{k,n}$); and 5) target warehouse.

A detailed example is as follows. If Prc_k contains ST_n, $R_{i,j}$ is equipped in MC_i, and ST_n in T_k is scheduled to $R_{i,j}$ by F, we set allocation sign $U_{k,n,i,j} = 1$, and 0 otherwise. Based on Figs. 2 and 3, we have $N_P = 2$, $N_{\mathrm{ST}} = 5$, and $N_R = 4$. The process types for task_0, task_1, and task_2 are $\mathrm{process}_0$, $\mathrm{process}_1$, and $\mathrm{process}_1$, respectively, and the target warehouses for them are WH_0, WH_1, and WH_1, respectively. Given four MCs ($N_M = 4$) and two WHs ($N_W = 2$), an instance of subtask scheduling is shown in Fig. 4, and the subtasks from different tasks are shown with different styles of oval frame. The ST_0 of task_0 is handled in $R_{0,0}$, i.e., $U_{0,0,0,0} = 1$, the ST_2 of task_1 is handled in $R_{2,2}$, i.e., $U_{1,2,2,2} = 1$, and so on. Therefore, there exists a tree structure of relevant robots for each task in a given scheduling scheme, e.g., the robot-tree for task_0 in Fig. 5.

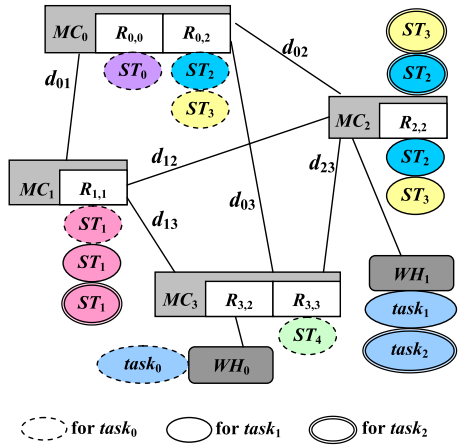


Fig. 4. Example of subtask scheduling.

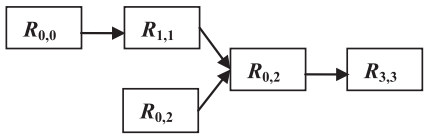


Fig. 5. Example of robot-tree for handling subtasks in task_0.

C. Robot Deployment Methods

Given predetermined types and number of robots, there are many methods for deploying robots in MCs, and the most common method is random deployment (labeled as **Ran-DP**). For controllable deployment of robots, there are several other typical methods.

1) Robot-balanced deployment (labeled as **RB-DP**) tries to allocate the types and numbers of robots evenly among MCs. The detailed steps are as follows.
   - Step 1) Count the number of robots for each type, and sort the robot types in an array by decreasing orders of these numbers.
   - Step 2) In each iteration, select the forefront robot type in this array, and allocate all robots of this type to all MCs evenly. The MCs with fewer robot types and numbers have the priority to get these robots.
   - Step 3) Remove the allocated robot type in this array, and go to step 2 for a new iteration.
2) Function-balanced deployment (labeled as **FB-DP**) tries to allocate functions and robot numbers evenly among MCs. The detailed steps are as follows.
   - Step 1) Count the number of function types for each robot type, and sort the robot types in an array by decreasing orders of these numbers.
   - Step 2) In each iteration, select the forefront robot type in this array, and allocate all robots of this type to all MCs evenly. The MCs with fewer function types and robot numbers have the priority to get these robots.
   - Step 3) Remove the allocated robot type in this array, and go to step 2 for a new iteration.

3) Location-aware deployment (labeled as **LA-DP**) tries to avoid the circuitous path for transportation. According to the process tree, the subtasks at the early stages of the process are handled in MCs that are farther away from target WH, and the subtasks at the late stages of the process are handled in MCs that are nearer to target WH. The detailed steps are as follows.

   Step 1) According to F, the average layer position for all subtasks (functions) in robot type j is

$$\mathrm{RL}_j = \frac{\sum_{n=0}^{N_{\mathrm{ST}}-1} F[j][n]\mathrm{STL}_n}{\sum_{n=0}^{N_{\mathrm{ST}}-1} F[j][n]}. \quad (2)$$

   RL_j embodies the relative order of employment of robot type j in production, and the RL_j set is sorted into array $\mathrm{RArray}[N_R]$ by decreasing order.

   Step 2) The average distance from MC_i to each WH is

$$\mathrm{MCD}_i = \frac{\sum_{j=0}^{N_W-1} D[i][j]}{N_W}. \quad (3)$$

   The MCD_i set is sorted into array $\mathrm{MCArray}[N_M]$ by decreasing order.

   Step 3) Allocate the elements in RArray to the elements in MCArray by the orders of elements in both arrays, and ensure that each MC consists of about N_R/N_M robots.

IV. Optimal Scheduling Strategies and Implementation

A. Optimization Strategies

A subtask can be handled by many robot types, so various scheduling schemes exist and yield different performances. The following are three strategies with different optimization objectives [17].

1) LBS: When ST_n of T_k is scheduled to $R_{i,j}$, we get Prc_k and the unit production output for ST_n in Prc_k. The actual output for ST_n of T_k is the product of this unit production output and TM_k, and the load of $R_{i,j}$ (labeled as $L_M(i,j)$) is represented by the sum of outputs for all subtasks in it

$$L_M(i,j) = \sum_{k=0}^{N_T-1}\sum_{n=0}^{N_{\mathrm{ST}}-1} U_{k,n,i,j}\mathrm{TM}_k\mathrm{PV}_{\mathrm{Prc}_k,n}. \quad (4)$$

In load-balance scheduling (LBS), the optimization objective is to minimize the standard deviation of the loads for all $R_{i,j}$

$$\begin{aligned}&\text{Minimize } (\mathrm{Std}\,(L_M(i,j)))\\ &\text{s.t.}\quad 0 \le i < N_M, 0 \le j < N_R.\end{aligned} \quad (5)$$

2) CMS: The overall cost for $R_{i,j}$ (labeled as $C(i,j)$) consists of production cost $C_M(i,j)$ and transportation cost $C_T(i,j)$ for all subtasks in $R_{i,j}$, i.e.,

$$C(i,j) = C_M(i,j) + C_T(i,j). \quad (6)$$

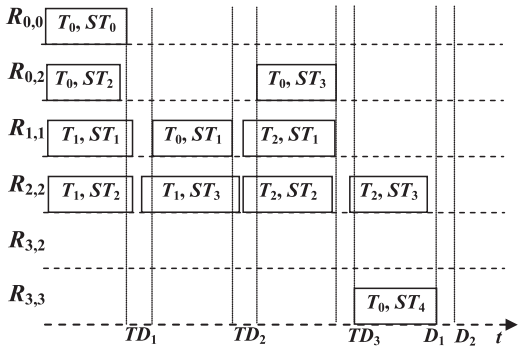


Fig. 6. Gantt chart of subtask handling.

When ST_n of T_k is scheduled to $R_{i,j}$, the production cost for ST_n of T_k is the product of the actual output for ST_n of T_k and $\mathrm{PC}_{i,j}$. Moreover, $C_M(i,j)$ is the sum of these subtasks' production costs in $R_{i,j}$

$$C_M(i,j) = \sum_{k=0}^{N_T-1}\sum_{n=0}^{N_{\mathrm{ST}}-1} U_{k,n,i,j}\mathrm{TM}_k\mathrm{PV}_{\mathrm{Prc}_k,n}\mathrm{PC}_{i,j}. \quad (7)$$

The transportation cost for ST_n of T_k is the product of TC_i, transportation distance from current MC to the next MC, and the actual transportation volume. Moreover, $C_T(i,j)$ is the transportation cost for all subtasks in $R_{i,j}$

$$C_T(i,j) = \sum_{k=0}^{N_T-1}\sum_{n=0}^{N_{\mathrm{ST}}-1} U_{k,n,i,j}\mathrm{TM}_k\mathrm{TV}_{\mathrm{Prc}_k,n} \times \mathrm{TC}_i D(i, \mathrm{Pos}_{k,\mathrm{Prn}_{k,n}}). \quad (8)$$

In cost-minimized scheduling (CMS), the optimization objective is to minimize the overall cost for all $R_{i,j}$

$$\begin{aligned}&\text{Minimize } \left(\sum_{i=0}^{N_m-1}\sum_{j=0}^{N_R-1} C(i,j)\right)\\ &\text{s.t.}\quad 0 \le i < N_M, 0 \le j < N_R.\end{aligned} \quad (9)$$

3) DMS: The processing time of a subtask is composed of the production time in a robot and the transportation time to the next MC or WH for the resultant component or product. Moreover, the overall production time for these tasks is determined by the subtask that finishes last. There are fast and convenient logistics services for the regional enterprise cluster, and the transportation can be carried on simultaneously while the robots are handling subtasks, so the overall production time accounts for a major part of the overall processing time.

Based on the scheduling scheme in Fig. 4, we give a demonstration of distributed subtask handling in Fig. 6. According to the process of T_0, there is a delay TD_1 for transporting the component of ST_0 to $R_{1,1}$, and then, ST_1 in T_0 can start. Similar delays (TD_2 and TD_3) are incurred for ST_3 and ST_4 in T_0 to start. Although there is only one subtask in $R_{3,3}$, it finishes work last at D_1, and all remain transportation finish at D_2.

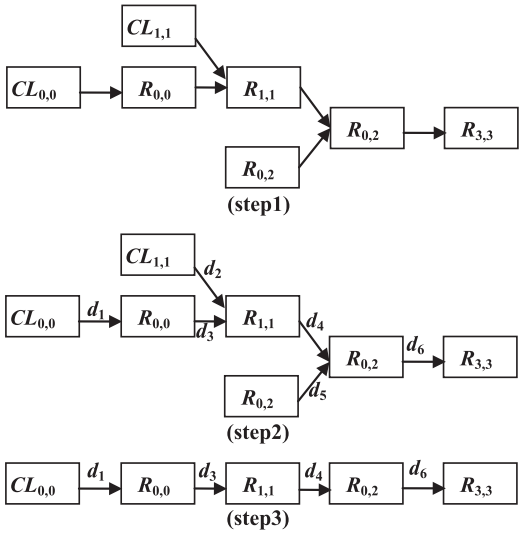


Fig. 7. Example for calculating the overall processing time for a task.

There are dependent relations among subtasks in a process, and the current subtask cannot be handled until the relevant earlier stage subtasks in the process are handled by other robots. For example, the only subtask ST_4 of T_0 in $R_{3,3}$ should wait for the completion of all of its previous subtasks, and the overall production time for $R_{2,2}$ is smaller than that for $R_{3,3}$. Therefore, we cannot calculate the overall production time from the view of a single robot. From the view of task, the delay-minimized scheduling (DMS) exists when all of the subtasks in the task that finishes last are handled without discontinuous idle waiting. In this case, the processing time of T_k is the sum of time for handling individual subtasks in different robots plus the necessary transportation time. Based on Fig. 5, we demonstrate the steps for calculating the shortest processing time in Fig. 7.

Step 1) **Robot-tree construction**. If there is a remaining workload to be processed in a robot, we add a node indicating current load to the corresponding node in robot-tree (e.g., $R_{0,0}$ and $R_{1,1}$ in Fig. 7) and thus get an expanded robot-tree.

Step 2) **Weighing the edges of robot-tree**. Given a directional edge from out-node to in-node in robot-tree, we calculate the sum d_i ($1 \leq i \leq 6$ in Fig. 7) of the time for handling the relevant subtask with out-node and that for transportation to in-node. Moreover, d_i is regarded as the weight of this edge.

Step 3) **Robot-tree trimming**. Considering the parallel processing of subtasks for a task, we select the trunk with the highest weight in robot-tree. The optimal processing time $D(k)$ for T_k is the sum of edge's weight in the trunk. For example, in Fig. 7, the trunk for task_0 consists of four edges, and $D(0)$ of current scheduling scheme is $d_1 + d_3 + d_4 + d_6$.

Considering that only one subtask can be handled first in a robot, we cannot achieve the optimal scheduling for all tasks and focus on the task that finishes last with its optimal scheduling scheme. In DMS, the minimal overall processing time that is possible to achieve is the maximum of all $D(k)$, and the optimization objective is to minimize it in the system

$$\begin{aligned} &\text{Minimize } (\text{Max } (D(k))) \\ &\text{s.t.} \quad 0 \leq k < N_T. \end{aligned} \tag{10}$$

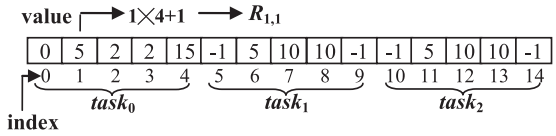


Fig. 8. Example of chromosome encoding.

B. Subtask Scheduling Inside Robots

For the general case, the minimal overall production time in a robot is achieved by consecutively handling the subtask with the shortest processing time first [18]. However, in CMF, there are complex dependent relations among subtasks from different robots, so the subtask with short processing time may not be handled first. Moreover, the minimal processing time of one robot cannot lead to the optimal performance for the whole system. For the same reason, ordering these subtasks in the robot by the deadlines of the corresponding tasks is also unreasonable. The STL_n in the process tree embodies the relative order of subtask handling, so we adopt the tactic that sorts the subtasks in a robot by the decreasing order of their STL_n.

C. Implementation of Approximate Optimal Scheduling

The scheduling can be described as an optimal assignment problem for bipartite graph, where some vertices are subtasks and others are robots, and each robot can be assigned with more than one subtask, so it is an NP-hard problem [19]. There are many algorithms for this assignment problem, including GA, ACO [20], and heuristic algorithm [21], and it is proved that GA can search the solution space effectively and evaluate many solutions simultaneously for avoiding locally optimal solution [22], so this paper adopts GA to solve this problem. The algorithm starts by encoding the problem to produce a list of genes. The genes are then randomly combined to produce a population of chromosomes, each of which represents a possible scheduling scheme. Iterative genetic operations for offspring are performed on chromosomes that are randomly selected from the population. The probability of chromosomes' survival is determined by their fitness values. As a self-adaptive and multidirectional searching process for optimization, it can operate on the final results directly with satisfactory robustness. The detailed stages are the following.

1) **Chromosome encoding**: In each chromosome, the genes are indexed by $k * N_{\mathrm{ST}} + n$, representing the ST_n in T_k, and are assigned with $i * N_R + j$, representing the $R_{i,j}$ that will handle ST_n in T_k. If ST_n is not in T_k, the gene is assigned with a negative value -1. The chromosome for the scheme in Fig. 4 is shown in Fig. 8.
2) **Generation of initial population**: Based on F, assign each gene with a robot that is capable of handling this

subtask randomly, and get N chromosomes as the initial population. Generally, N is within the range [40, 120].

3) **Fitness evaluation**: We define three fitness functions for the aforementioned optimization objectives as follows:

$$\text{Fit}_1 = \frac{1}{\text{Std}\left(L_M(i,j)\right)} \tag{11}$$

$$\text{Fit}_2 = \frac{1}{\sum_{j=0}^{N_R-1} C(i,j)} \tag{12}$$

$$\text{Fit}_3 = \frac{1}{\text{Max}\left(D(k)\right)}. \tag{13}$$

4) **Chromosome selection**: Based on the fitness threshold, excellent chromosomes are selected from parent generation. Here, we adopt the proportional selection method [19], which takes the ratio of the fitness value for a chromosome to the sum of fitness values for all chromosomes as the selection probability.
5) **Chromosome crossover**: It is the most vital operation for new generation and embodies the idea of information exchange. We make crossover for the gene segment of each task and select the starting position for crossover in each segment randomly.
6) **Chromosome mutation**: With a random small probability, we select some genes among all chromosomes and change their values to embody the effect of mutation.
7) **New population**: After the aforementioned stages, we evaluate the fitness values for all chromosomes, select N chromosomes with the highest fitness values as the new population, and go to stage 4) for the next iteration. When the algorithm converges, we terminate the algorithm and output the chromosome with the highest fitness value as the selected scheduling scheme.

V. Numerical Analysis

A. Setting of Simulation Parameters

Taking a CMF instance of heavy-machinery production as an example, we get the optimal chromosomes under different optimization strategies and deployment methods in MATLAB and conduct simulation experiments based on the scheduling schemes from these chromosomes.

We derive the fixed parameters from the aforementioned example for simulation as follows: $N_M = 8$, $N_R = 10$, $N_W = 4$, $N_{\text{ST}} = 15$, and $N_P = 15$, and the total number of robots in the system is 40. The parameters in Table II are assigned with integer values within a given range to embody the diversity of the parameters' values.

Other special parameters are as follows. The process trees are constructed according to a relatively fixed order of subtasks handling in the process, so we specify that five subtask types exist in each layer of the process tree with equal probability, five subtask types exist in the top third layers of the process tree with a probability of 70%, and five subtask types exist in the bottom third layers of the process tree with a probability of 70%. $\text{TV}_{m,n}$ is within range of [1], [7] ton. Moreover, to embody the effect of component assembling and weight increasing as the process goes, $\text{TV}_{m,n}$ is larger for the subtasks that are nearer to the root of the process tree.

TABLE II
Parameter Setting for Simulation

Parameter	Value range
$D(u,v)$	[500,1500] km
TC_i	[60,80] \$/(ton·100km)
$PC_{i,j}$	[20,40] \$
$PD_{i,j}$	[30,60] minutes
$PV_{m,n}$	[1,5] times of ERC
TM_k	[50,100]
number of subtasks in a process	[6,9]
number of functions for each robot type	[1,5]

B. Setting of Simulation Scenarios

For the creditability of the simulation results, we assign types to all 40 robots randomly and get 20 different robot-type allocation scenarios labeled as rs_p $(1 \leq p \leq 20)$. By assigning N_T with numbers from 15, 20, 25, . . ., 60, we get 10 task scenarios labeled as ts_q $(1 \leq q \leq 10)$ for different workloads. To get the global approximate optimal solutions of the GA algorithm, we conduct simulation experiments under these scenarios, so we can overcome the possible deficiency from fitness function and other parameters in population initialization, crossover method, and mutation method.

In each two-tuple of $(\text{rs}_p, \text{ts}_q)$, we provide the same initial chromosomes and parameters for all scheduling strategies and deployment methods and get the corresponding optimal scheduling schemes. The GA algorithm converges when there is no obvious difference (less than 0.5%) of fitness values for quite a few generations. The number of generations for algorithm converging varies from 50 to 120.

For comparison of scheduling strategies under Ran-DP, we conduct simulation experiment with relevant schemes and get the performance indexes such as standard deviation of robot load, overall cost, and overall processing time. For comparison of deployment methods, we conduct a simulation experiment with relevant schemes and get the performance indexes such as the standard deviation of the robot load by LBS, overall production cost by CMS, overall transportation cost by CMS, and overall processing time by DMS. At last, we get the average values of these performance indexes for all 20 rs_p to eliminate the influence of noise data and plot these results under each ts_q in Figs. 9 and 10.

C. Analysis on Scheduling Strategies

From Fig. 9, we find that each scheduling strategy can get the optimal performance for the corresponding optimization objective.

In Fig. 9(a), LBS yields the minimal standard deviation of the robot load. Subtasks tend to amass in robots with low cost for CMS and amass in robots with small handling delay for DMS,

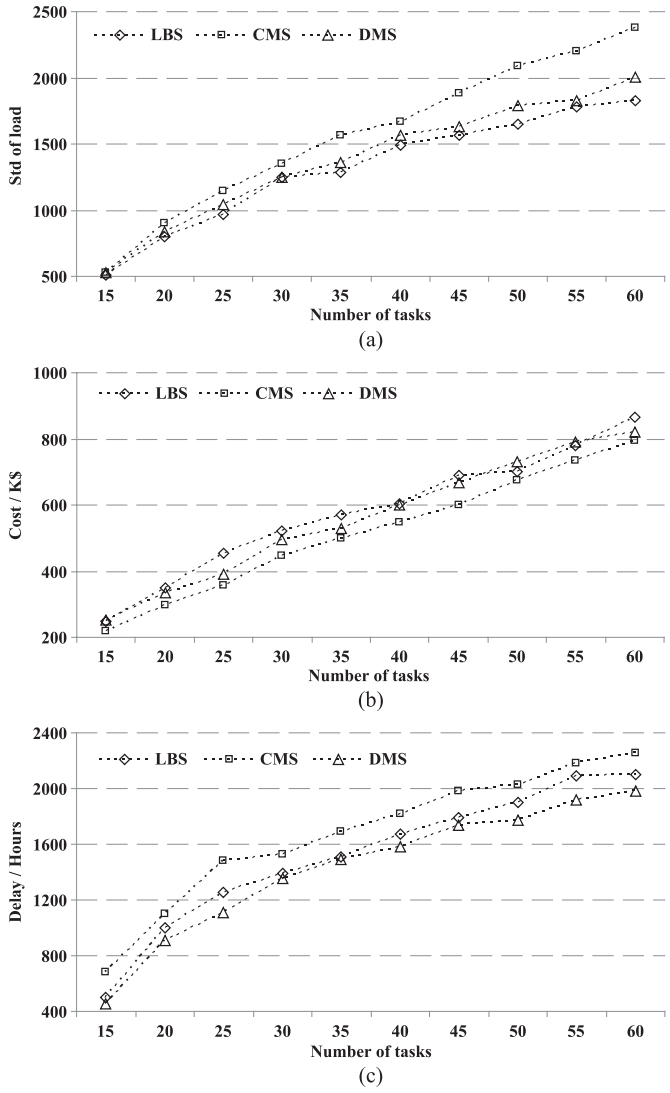


Fig. 9. Performance of different scheduling strategies. (a) Standard deviation of the robot load. (b) Overall cost. (c) Overall processing time.

so the two scheduling strategies lead to more severe imbalance of processing load in robots. As the workload increases, subtasks are more evenly allocated to robots, and the growth rate of the standard deviation of the load keeps decreasing in LBS. On the other hand, the imbalance of the load in CMS is much more severe than those in the other two strategies.

In Fig. 9(b), the production costs for DMS and LBS are higher than that for CMS. The overall cost grows as the workload increases for all three scheduling strategies. Although subtasks amass in a few robots of low production cost, the distance and cost of transportation for these subtasks may also increase in the same time, so the growth rate of the overall cost between two ts_j's does not drop noticeably as the workload increases.

In Fig. 9(c), the overall processing time for DMS is the lowest. When the workload increases to a certain level (e.g., $N_T = 25$), the growth rate of the production time between two ts_j's drops noticeably. It illustrates that, when the workload is high, there is severe congestion of subtasks in robot, so the production time overwhelms the transportation time, and the overall processing time is mostly composed of production time.

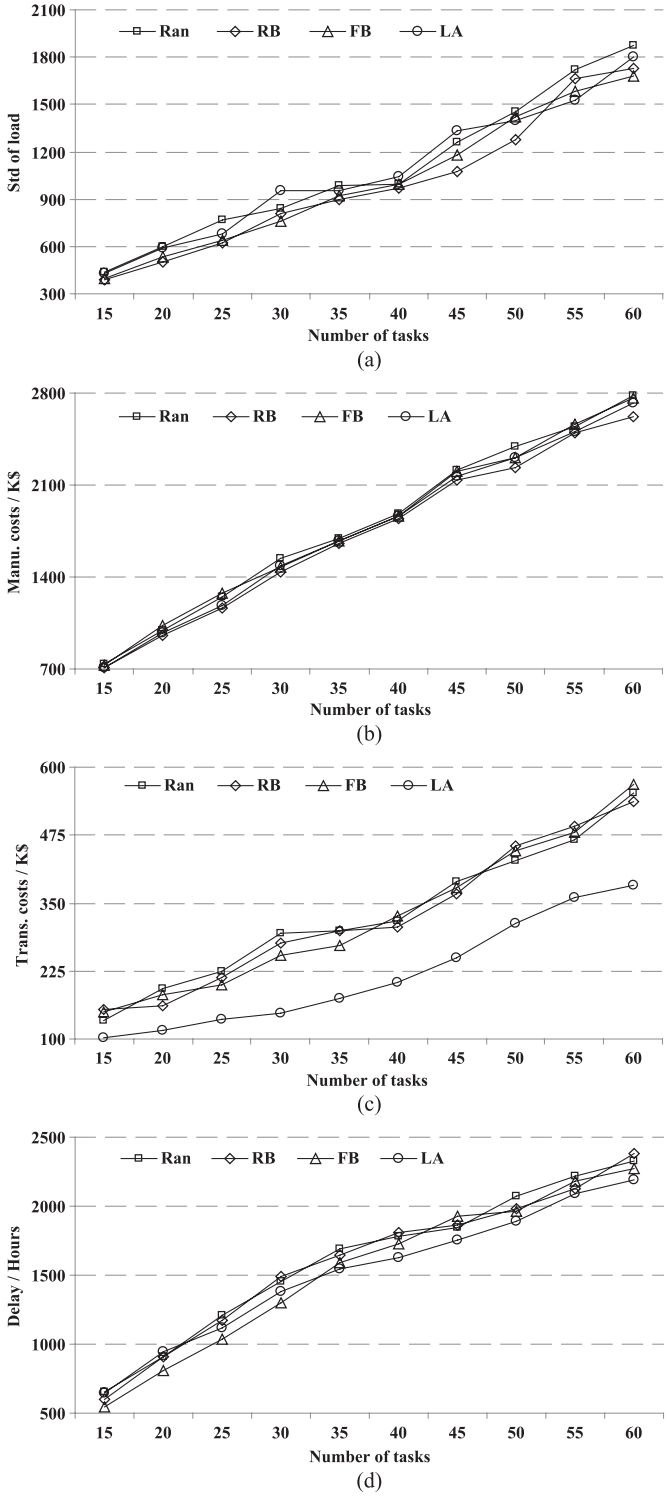


Fig. 10. Performance of different deployment methods. (a) Load-balance of robots. (b) Overall production cost. (c) Overall transportation cost. (d) Overall processing time.

D. Analysis on Robot Deployment Methods

In Fig. 10(a), each deployment method with LBS yields similar performances of load-balance with one another. Moreover, in Fig. 10(b), each deployment method with CMS yields similar overall production costs with one another. Therefore, we conclude that there is no obvious difference among these deployment methods on these performance indexes.

In Fig. 10(c), LA-DP yields a much smaller transportation cost than the other three methods. However, there is no much difference on transportation costs for the other three deployment methods. The reason is that LA-DP can decrease the circuitous transportation effectively, and as subtasks in the process tree are handled in order, their components are getting nearer to target WH. Considering that transportation cost plays a minor role in overall cost, we conclude that LA-DP shows a slight advantage over other deployment methods on overall cost.

In Fig. 10(d), under the low-workload state when N_T is small, LA-DP leads to the minimal overall processing time. The reason is the shortened transportation distance by LA-DP. On the other hand, under the high-workload state, transportation has little influence on overall processing time, and FB-DP leads to the minimal overall processing time. The reason is that FB-DP provides each MC with as many functions as possible, so many subtasks of the same task can be handled in one MC, and there is less dependence on robots in other MCs. Similarly, high-workload leads to congestion of subtasks, so the growth rate of the overall processing time between two ts_j's drops noticeably for the decreased influence of transportation time.

VI. Conclusion

For distributed subtask handling in CMF platform, this paper has proposed a CMF scheduling model for cooperative manufacturing in regional enterprise cluster, together with several scheduling optimization strategies and several robot deployment methods. Simulation results have shown that LBS, CMS, and DMS yield optimal performances of load-balance in robots, overall cost, and overall processing time, respectively. Physical distance may take effect on overall cost, and LA-DP yields the minimal transportation cost. LA-DP and FB-DP yield the minimal overall processing time in low-workload and high-workload states, respectively. High workload leads to congestion of subtasks in robots, so the production time occupies larger portion in the overall processing time as the workload increases.

Based on the aforementioned insights, further valuable research directions include the following: 1) comprehensive performance optimization index and deployment methods with considerations of varying production costs and delay, logistics to users, and inventory costs; 2) scheduling the subtasks in a robot with considerations of priority and arriving pattern of task; and 3) in a view of complex networks, exploring the features of robot-trees for dynamically increasing tasks with different robot-selection preferences.

References

[1] R. Doriya, P. Chakraborty, and G. C. Nandi, "Robotic services in cloud computing paradigm," in *Proc. IEEE Int. Symp. Cloud Serv. Comput.*, 2012, pp. 80–83.

[2] D. Z. Wu *et al.*, "Cloud manufacturing: Strategic vision and state-of-the-art," *J. Manuf. Syst.*, vol. 32, no. 4, pp. 564–579, Oct. 2013.

[3] L. N. Zhu, Y. W. Zhao, and W. L. Wang, "A bilayer resource model for cloud manufacturing services," *Math. Problems Eng.*, vol. 2013, 2013, Art. ID. 607582.

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

[5] C. L. Huang and C. C. Huang, "Cloud computing based intelligent manufacturing scheduling system using the quality prediction method," *Trans. Can. Soc. Mech. Eng.*, vol. 37, no. 3, pp. 981–989, 2013.

[6] Z. H. Du *et al.*, "Design of a robot cloud center," in *Proc. IEEE 10th Int. Symp. Auton. Decentralized Syst.*, 2011, pp. 269–275.

[7] Y. J. Laili, L. Zhang, and F. Tao, "Energy adaptive immune genetic algorithm for collaborative design task scheduling in cloud manufacturing system," in *Proc. IEEE Int. Conf. Ind. Eng. Eng. Manage.*, 2011, pp. 1912–1916.

[8] Y. J. Laili, F. Tao, L. Zhang, and L. Ren, "The optimal allocation model of computing resources in cloud manufacturing system," in *Proc. IEEE 7th Int. Conf. Nat. Comput.*, 2011, vol. 4, pp. 2322–2326.

[9] Y.-W. Zhao and L.-N. Zhu, "Service evaluation-based resource selection in cloud manufacturing," in *Cooperative Design, Visualization, and Engineering*, vol. 8683, Lecture Notes in Computer Science. Cham, Switzerland: Springer-Verlag, 2014, pp. 294–302.

[10] T. K. Jana, B. Bairagi, S. Paul, B. Sarkar, and J. Saha, "Dynamic schedule execution in an agent based holonic manufacturing system," *J. Manuf. Syst.*, vol. 32, no. 4, pp. 801–816, Oct. 2013.

[11] A. Benoit, A. Dobrila, J.-M. Nicod, and L. Philippe, "Scheduling linear chain streaming applications on heterogeneous systems with failures," *Future Gener. Comput. Syst.*, vol. 29, no. 5, pp. 1140–1151, Jul. 2013.

[12] M. Wang, J. Zhou, and S. Jing, "Cloud manufacturing: Needs, concept and architecture," in *Proc. IEEE 16th Int. Conf. Comput. Supported Cooperative Work Design*, 2012, pp. 321–327.

[13] W. Liu, B. Liu, D. Sun, and G. Ma, "Study on multi-task oriented services composition and optimisation with the "multi-composition for each task" pattern in cloud manufacturing systems," *Int. J. Comput. Integr. Manuf.*, vol. 26, no. 8, pp. 786–805, Aug. 2013.

[14] P. Ge *et al.*, "Task allocation in cloud manufacturing based on multi-resolution clustering," (in Chinese), *Comput. Integr. Manuf. Syst.*, vol. 18, no. 7, pp. 1461–1468, 2012.

[15] L. Jorick, L. Nie, X. Xu, D. Zhan, and T. Mou, "Scheduling methodology for production services in cloud manufacturing," in *Proc. Int. Joint Conf. Serv. Sci.*, 2012, pp. 34–39.

[16] Y. Zhang, Z. Ni, Y. Jin, and H. Li, "Large-scale partner selection of virtual enterprise on the cloud manufacturing service platform," *J. Inf. Comput. Sci.*, vol. 9, no. 14, pp. 3995–4006, 2012.

[17] M. L. Pinedo, *Scheduling: Theory, Algorithms and Systems*. New York, NY, USA: Springer Science and Business Media, LLC, 2008, pp. 59–67.

[18] G. Faruk, "Process plan and part routing optimization in a dynamic flexible job shop scheduling environment: An optimization via simulation approach," *Neural Comput. Appl.*, vol. 23, no. 6, pp. 1631–1641, Nov. 2013.

[19] I. H. Toroslu and Y. Arslanoglu, "Genetic algorithm for the personnel assignment problem with multiple objectives," *Inf. Sci.*, vol. 177, no. 3, pp. 787–803, Feb. 2007.

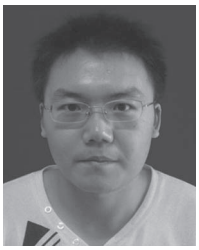
[20] D. Djamarus and K. R. Ku-Mahamud, "Heuristic factors in ant system algorithm for course timetabling problem," in *Proc. IEEE 9th Int. Conf. Intell. Syst. Design Appl.*, 2009, pp. 232–236.

[21] A. Abdullah, "Data Mining Using the Crossing Minimization," Ph.D. dissertation, Dept. Comput. Sci. Math., Univ. Stirling, Stirling, U.K., 2007.

[22] S. Khan, M. Bilal, M. Sharif, and F. A. Khan, "A solution to bipartite drawing problem using genetic algorithm," in *Advances in Swarm Intelligence*, vol. 6728, Lecture Notes in Computer Science. Berlin, Germany: Springer-Verlag, 2011, pp. 530–538.

Wenxiang Li (M'14) was born in 1979. He received the B.E. degree in communication engineering, the M.E. degree, and the Ph.D. degree in communication and information system from Wuhan University, Wuhan, China, in 2001, 2004, and 2010, respectively.

He is currently an Associate Professor with Wuhan University of Science and Technology, Wuhan. He was a Visiting Professor with The University of British Columbia, Vancouver, BC, Canada. He has published more than 40 papers in different journals and conferences. His research interests include cognitive wireless networks and cloud computing.



Chunsheng Zhu (S'12) received the B.E. degree in network engineering from Dalian University of Technology, Dalian, China, in 2010 and the M.Sc. degree in computer science from St. Francis Xavier University, Antigonish, NS, Canada, in 2012. He has been working toward the Ph.D. degree with Four-Year-Fellowship in the Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, BC, Canada, since September 2012.

He has around 40 papers published or accepted by refereed international journals (e.g., IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTING, and IEEE SYSTEMS JOURNAL) and conferences (e.g., IEEE Globecom and IEEE ICC). His current research interests are mainly in the areas of wireless sensor networks and mobile cloud computing.

Laurence T. Yang (M'97) received the B.E. degree in computer science and technology from Tsinghua University, Beijing, China, and the Ph.D. degree in computer science from the University of Victoria, Victoria, BC, Canada.

He is a Professor with the Department of Computer Science, St. Francis Xavier University, Antigonish, NS, Canada. His research interests include parallel and distributed computing, embedded and ubiquitous/pervasive computing, and big data. He has published more than 220 papers in various refereed journals (around 40% in top IEEE/ACM Transactions and Journals, others mostly in Elsevier, Springer, and Wiley Journals). His research has been supported by the National Sciences and Engineering Research Council and the Canada Foundation for Innovation.

Lei Shu (M'07) received the B.Sc. degree in computer science from South Central University for Nationalities, Wuhan, China, in 2002, the M.Sc. degree in computer engineering from Kyung Hee University, Seoul, Korea, in 2005, and the Ph.D. degree from the Digital Enterprise Research Institute, National University of Ireland, Galway, Ireland, in 2010.

He is currently a Full Professor with the College of Electronic Information and Computer, Guangdong University of Petrochemical Technology, Maoming, China, and also the Vice-Director of the Guangdong Petrochemical Equipment Fault Diagnosis (PEFD) Key Laboratory. He was a Specially Assigned Research Fellow with the Department of Multimedia Engineering, Graduate School of Information Science and Technology, Osaka University, Suita, Japan. He has published over 200 papers in related conferences, journals, and books. His research interests include wireless sensor networks, sensor network middleware, multimedia communication, and security.

Dr. Shu is a member of ACM. He received the Globecom 2010 and ICC 2013 Best Paper Awards. He is the Editor-in-Chief of *EAI Endorsed Transactions on Industrial Networks and Intelligent Systems*, and he has served as editor and guest editor of more than 20 international journals. He has served as the Cochair for more than 50 various international conferences/workshops, e.g., IWCMC, ICC, and ISCC; and TPC member of more than 150 conferences, e.g., ICC, Globecom, ICCCN, WCNC, and ISCC. He has served as a reviewer of more than 50 journals.

Edith C.-H. Ngai (M'01) received the Ph.D. degree from The Chinese University of Hong Kong, Hong Kong, in 2007.

She is currently an Associate Professor with the Department of Information Technology, Uppsala University, Uppsala, Sweden. She was a Postdoc with the Imperial College London, London, U.K., in 2007–2008. Previously, she has conducted research at Simon Fraser University, Burnaby, BC, Canada, Tsinghua University, Beijing, China, and University of California, Los Angeles, CA, USA. Her research interests include wireless sensor and mobile networks, Internet-of-things, network security and privacy, smart city, and e-health applications.

Dr. Ngai is a VINNMER Fellow (2009) awarded by VINNOVA, Sweden. Her coauthored papers have received best paper runner-up awards at IWQoS 2010 and IPSN 2013. She has served as a TPC member of IEEE ICDCS, IEEE ICC, IEEE Globecom, IEEE/ACM IWQoS, IEEE DCOSS, IEEE LCN, etc. She served as a TPC Cochair of Swedish National Computer Networking Workshop (SNCNW'12) and QShine'14. She is a Program Cochair of ACM womENcourage 2015, TPC Cochair of SmartCity 2015, and ISSNIP 2015.

Yajie Ma (M'13) was born in 1974. She received the B.E. degree from Wuhan University of Science and Technology, Wuhan, China, in 1996 and the M.E. and Ph.D. degrees from Huazhong University of Science and Technology, Wuhan, in 2000 and 2005, respectively.

She held her postdoctoral research work at the Data-Mining Research Group, London Imperial College, London, U.K., from 2006 to 2009. She is currently a Professor with Wuhan University of Science and Technology. She has published more than 30 papers in different journals and conferences. Her research interests include wireless sensor networks and distributed data mining.