Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 1305849, 10 pages
https://doi.org/10.1155/2021/1305849

Research Article

Multiobjective Real-Time Scheduling of Tasks in Cloud Manufacturing with Genetic Algorithm

Gilseung Ahn and Sun Hur

Department of Industrial and Management Engineering, Hanyang University, Ansan 15588, Republic of Korea

Correspondence should be addressed to Sun Hur; hursun@hanyang.ac.kr

Received 20 April 2020; Accepted 21 September 2020; Published 15 February 2021

Academic Editor: Manjit Kaur

In cloud manufacturing, customers register customized requirements, and manufacturers provide appropriate services to complete the task. A cloud manufacturing manager establishes manufacturing schedules that determine the service provision time in a real-time manner as the requirements are registered in real time. In addition, customer satisfaction is affected by various measures such as cost, quality, tardiness, and reliability. Thus, multiobjective and real-time scheduling of tasks is important to operate cloud manufacturing effectively. In this paper, we establish a mathematical model to minimize tardiness, cost, quality, and reliability. Additionally, we propose an approach to solve the mathematical model in a real-time manner using a multiobjective genetic algorithm that includes chromosome representation, fitness function, and genetic operators. From the experimental results, we verify whether the proposed approach is effective and efficient.

1. Introduction

Cloud manufacturing (CM) is an advanced model that provides a cloud-based manufacturing platform on which enterprises virtualize and share their manufacturing services such as welding, milling, and machining to produce highly customized products [1–3]. A task to produce a customized product consists of several activities, and each activity provides a related manufacturing service. The task is processed in the following order [4]. First, a platform manager determines the manufacturing services and an appropriate processing order. This step is called a task definition. The manager then establishes a schedule by selecting services from the platform and determines the start times of each service, considering availability, processing time, and service order. This step, called task scheduling, differs from service allocation in that it considers time [5]. In other words, it allows duplicate assignment of a service (in this case, delay may occur), while service allocation does not. Finally, the task sequence begins processing according to the schedule.

Because task scheduling is an important operational problem in CM, many researchers have recently addressed it. Zhou et al. [6] modeled a scheduling problem to minimize the weighted sum of differences between requirement levels and actual levels in terms of time, cost, and quality, and service type is considered a major constraint. They improved a genetic algorithm (GA) to solve the scheduling problem and compared it with particle swarm optimization (PSO) and simulated annealing in an experiment in which each task was given a different weight. They argued that their proposed algorithm outperformed other algorithms in terms of objective value. Cao et al. [7] investigated a task-scheduling problem in terms of time, quality, cost, and service and adopted fuzzy decision-making theory to transform these criteria into degrees of relative superiority. They presented a mathematical model that maximizes the weighted sum of the relative degrees and applied ant colony optimization (ACO) to obtain a solution. Through experiments, they showed that ACO outperformed GA and PSO in every iteration. Liu et al. [8] revealed that workloads affect total completion time, service utilization, and so forth in task-scheduling problems. They compared two scheduling methods based on workload. The first completes the tasks with larger workloads first, while the second completes tasks with smaller workloads first. They concluded that the first method yields a superior performance. Jiang et al. [9] addressed a task-scheduling problem in CM that specializes in disassembly, with the objective of minimizing both total expected makespan and cost. They designed a multiobjective algorithm using the nondominated sorting genetic algorithm II (NSGA-II) to solve the task-scheduling problem. Li et al. [10] studied a task-scheduling problem for distributed robots in CM in which tasks were scheduled by allocating a robot to a subtask, considering geographical location. They solved the problem from three perspectives: difference of workload among robots, overall cost, and overall processing time with a GA.

Although previous research investigated task scheduling from multiple perspectives, the chosen approaches were unrealistic in terms of multiobjective and real-time characteristics. Scheduling problems should consider two or more objectives simultaneously. Some researches such as in [6] considered multiple objectives as a weighted sum, but they also were not practical as it proved difficult to choose the proper weight. Only a few (e.g., [9]) considered simultaneous objectives, when modeling the scheduling problem. In addition, the proper services should be used in real-time process (sub) tasks whenever they are registered because this aspect is an important characteristic of CM platform operations.

Meta-heuristic algorithms have been frequently employed to solve computational engineering problems [11–17]. For example, Hoang [11] integrated history-based adaptive differential evolution and linear population size reduction to find the optimal hyperparameters of the support vector machine for pitting corrosion detection. Chen et al. [12] developed a hybrid of variable neighborhood search (VNS) and estimation of distribution algorithm (EDA), called VNS-EDA, to compose the feature subset for credit risk classification. Jiang et al. [18] improved particle swarm optimization (PSO) by integrating with the gravitational search algorithm with dependent random coefficients. Peng et al. [19] introduced chaotic search for the fuzzy neural network to improve PSO to handle complex engineering problems.

A GA is one of the most famous meta-heuristic algorithms to solve diverse scheduling problems such as project scheduling [20, 21], job-shop scheduling [22], parallel machine scheduling [23], and flow-shop scheduling [24, 25] and usually performs well. In addition, a multiobjective genetic algorithm (MOGA) has also been successfully applied to various multiobjective problems, such as a scheduling problem to minimize both production cost and time. For example, Tang et al. [26] addressed a multiobjective radio frequency identification network-planning problem with the objective of minimizing collision and interference of the network and network cost. They integrated a divide and conquer greedy heuristic algorithm and an MOGA to solve the problem. Zhang et al. [27] addressed multiobjective assembly line-balancing problems to minimize cycle time and rebalancing cost by modifying NSGA-II. Because GA is appropriate for handling the multiobjective scheduling problem, as shown in many previous researches, we choose GA to solve a multiobjective real-time task-scheduling problem in CM.

In this paper, we study a multiobjective real-time task-scheduling problem in CM where tasks are scheduled whenever they are registered. The major research contents and contributions are as follows. First, we formalized the scheduling problem as a binary integer programming model with four objectives to minimize total tardiness, cost, quality, and reliability penalties. In addition, we design an MOGA to solve the problem by focusing on feasibility because most solutions to the problem are infeasible. Finally, we conduct an experiment to verify the proposed approach's performance. Table 1 summarizes how the contribution of our study extends previous research results.

The rest of this paper is organized as follows. Section 2 describes the problem and introduces a mathematical model. Section 3 develops the proposed approach by focusing on designing a multiobjective GA that includes chromosome representation, a fitness function, and genetic operators. Section 4 conducts the experiments and compares the efficiency of the GA, and Section 5 concludes the paper.

2. Problem Description and Mathematical Model

2.1. Notation. Table 2 shows notations used in this paper.

2.2. Problem Description. In cloud manufacturing, customers upload their requirements in the cloud-based manufacturing platform, and then the requirements are converted into a task, each of which consists of several activities and saved in the task pool. Enterprises virtualize and upload their manufacturing services (e.g., milling and cutting), and the services are saved in the service pool. The task-scheduling problem is to make a schedule for each task in the task pool by assigning a set of proper manufacturing services in the service pool. The considered task-scheduling problem is to assign a proper service to each activity of every task to minimize total tardiness and cost, quality, and reliability penalties at each time. Figure 1 shows a typical example of the structure of a task-scheduling problem. The schedule is constructed by a manager (or management system) on a cloud-based manufacturing platform.

Each task is composed of several activities in sequence or in parallel. Task T_2^t, in Figure 1, for example, is composed of three activities: $A_{2,1}^t$, $A_{2,2}^t$, and $A_{2,3}^t$. ($A_{2,1}^t$, $A_{2,2}^t$) are composed sequentially, while ($A_{2,2}^t$, $A_{2,3}^t$) are in parallel. The required service type differs according to the activity. For example, if S_k is the service type needed by $A_{i,j}^t$, then one of $S_{k,v}$ ($v = 1, 2, \ldots, |S_k|$) can be matched to $A_{i,j}^t$. Each activity can be processed only after the preceding activities have been completed. The activity without preceding activities (the root activity) can be processed only after the task to which it belongs is registered. Finally, services may have different performance metrics such as cost and quality even though they are of the same type. Services should be scheduled considering all required levels of tasks.

Table 1: Contribution of this paper to previous research.

Previous study	Considering multiobjective	Considering real-time	Considering various composition types	Improving meta-heuristic algorithm
Zhou et al. [6]		√	√	√
Cao et al. [7]		√	√	√
Liu et al. [8]	√		√	
Jiang et al. [9]	√	√		√
Li et al. [10]	√		√	
This paper	√	√	√	√

Table 2: Notations.

Notation		Description
Service related	S_k	Type k service, $k = 1, 2, \ldots, l$
	$S_{k,v}$	v^{th} type k service, $v = 1, 2, \ldots, \lvert S_k \rvert$
	$c_{k,v}$	Cost of $S_{k,v}$
	$q_{k,v}$	Quality of $S_{k,v}$
	$r_{k,v}$	Reliability of $S_{k,v}$
	$p_{k,v}$	Processing time of $S_{k,v}$
Task and activity related	T_i^t	Task i registered at time $t = 1, 2, \ldots, T$, $i = 1, 2, \ldots, n_t$, where n_t is the number of tasks registered at time t
	$A_{i,j}^t$	Activity j in task T_i^t, $j = 1, 2, \ldots, m_i^t$, where m_i^t is the number of activities in T_i^t
	$\text{Type}(A_{i,j}^t)$	Service type required for $A_{i,j}^t$, $\text{type}(A_{i,j}^t) \in \{S_1, S_2, \ldots, S_l\}$
	$P_{i,j}^t$	Set of precedent activities of $A_{i,j}^t$
	$F_{i,j}^t$	Set of following activities of $A_{i,j}^t$
	D_i^t	Due date of T_i^t
	C_i^t	Maximum allowed cost to complete T_i^t
	Q_i^t	Minimum allowed quality of T_i^t
	R_i^t	Minimum allowed reliability of T_i^t
	$\widehat{D}_i^t$	Actual delivery date of T_i^t
	$\widehat{C}_i^t$	Actual cost to complete T_i^t
	$\widehat{Q}_i^t$	Actual quality of T_i^t
	$\widehat{R}_i^t$	Actual reliability of T_i^t
	$\widetilde{T}_i^t$	Tardiness penalty on T_i^t
	$\widetilde{C}_i^t$	Cost penalty on T_i^t
	$\widetilde{Q}_i^t$	Quality penalty on T_i^t
	$\widetilde{R}_i^t$	Reliability penalty on T_i^t
Decision variables	$x_{i,j,k,v}^{t,\tau}$	= 1, if $S_{k,v}$ is matched to $A_{i,j}^t$ at time $\tau \geq t$, and = 0, otherwise
	$I(S_{k,v}^t)$	= 1, if $S_{k,v}$ is available at time t, and = 0, otherwise

Additional assumptions of the problem are summarized as follows:

(1) Required service type to process each activity is known

(2) The maximum allowed cost, minimum allowed quality and reliability, and due date are given

(3) There is no rework for any activity

(4) Customer satisfaction can be measured by cost, quality, reliability, and due date

(5) Geographical distances among service providers are not considered

(6) The outcome of a task is delivered to the customer once the task is completed

2.3. Mathematical Model. We describe objective functions and constraints of the established mathematical model in Sections 2.3.1 and 2.3.2, respectively.

2.3.1. Objective Functions. The objectives are to minimize the total penalty associated with tardiness, cost, quality, and reliability. In this section, we explain how to obtain objective values from the service schedule.

Actual delivery time $\widehat{D}_i^t$ of task T_i^t is the maximum among completion times of activities that do not have the following activities. Because completion time of $A_{i,j}^t$ is calculated as the sum of service assignment time and processing time, $\widehat{D}_i^t$ is obtained as follows:

$$\widehat{D}_i^t = \max_{F_{i,j}^t = \varnothing} \left(\sum_{k = Type\left(A_{i,j}^t\right), \forall v, \tau \geq t} \left(x_{i,j,k,v}^{t,\tau} \times \left(\tau + p_{k,v} - 1 \right) \right) \right), \tag{1}$$

where $F_{i,j}^t$ and $p_{k,v}$ denote the set of $A_{i,j}^t$'s following activities and the processing time of $S_{k,v}$, respectively. With equation (1), the tardiness penalty $\widetilde{T}_i^t$ of T_i^t is computed as the difference

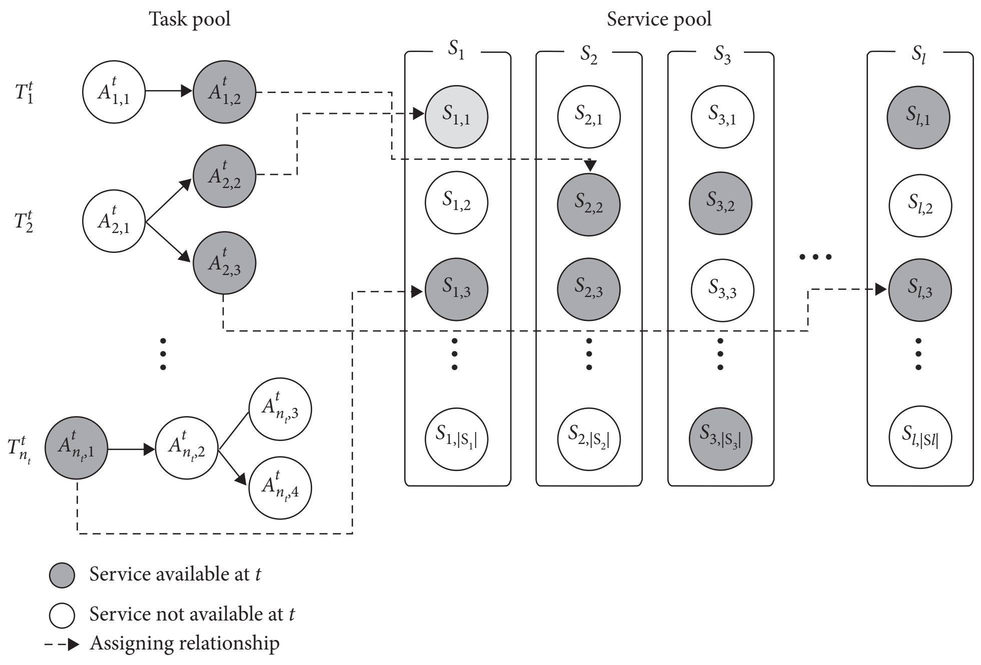


FIGURE 1: Typical example of task scheduling.

between $\widehat{D}_i^t$ and D_i^t if it is bigger than zero, and zero otherwise:

$$\widetilde{T}_i^t = \max\left(\widehat{D}_i^t - D_i^t, 0\right), \tag{2}$$

where D_i^t is the due date of T_i^t. By expanding equation (2) to all tasks registered at t, we have the first objective function, which is the sum of tardiness penalty for every task registered at t:

$$\text{minimize } Z_1 = \sum_{i=1}^{n_t} \widetilde{T}_i^t. \tag{3}$$

The actual cost $\widehat{C}_i^t$ to complete T_i^t is calculated as the sum of costs of services assigned to the task as given by

$$\widehat{C}_i^t = \sum_{j=1}^{m(t/i)} \sum_{k=Type\left(A_{i,j}^t\right),\forall v} \left(c_{k,v} \times x_{i,j,k,v}^{t,\tau}\right), \tag{4}$$

with equation (4), the cost penalty cp_i^t on T_i^t is

$$\widetilde{C}_i^t = \max\left(\widehat{C}_i^t - C_i^t, 0\right), \tag{5}$$

where C_i^t is the maximum allowed cost to complete T_i^t. By expanding equation (5) to all tasks, the second objective function is obtained as follows:

$$\text{minimize } Z_2 = \sum_{i=1}^{n_t} \widetilde{C}_i^t. \tag{6}$$

Similarly, the actual quality $\widehat{Q}_i^t$ of T_i^t is the average of qualities of services, and actual reliability $\widehat{R}_i^t$ is the product of reliabilities of services assigned to the task as given by

$$\widehat{Q}_i^t = \frac{\sum_{j=1}^{m(t/j)} \sum_{k=\text{Type}\left(A_{i,j}^t\right),\forall v} \left(q_{k,v} \times x_{i,j,k,v}^{t,\tau}\right)}{m_j^t},$$
$$\widehat{R}_i^t = \prod_{j=1}^{m(t/j)} \sum_{k=Type\left(A_{i,j}^t\right),\forall v} \left(r_{k,v} \times x_{i,j,k,v}^{t,\tau}\right), \tag{7}$$

where $\sum_{k=\text{Type}(A_{i,j}^t),\forall v} (r_{k,v} \times x_{i,j,k,v}^{t,\tau})$ is the reliability of the service assigned to activity $A_{i,j}^t$.

The quality penalty is $\widetilde{Q}_i^t = \max(Q_i^t - \widehat{Q}_i^t, 0)$, and the reliability penalty is $\widetilde{R}_i^t = \max(R_i^t - \widehat{R}_i^t, 0)$. The third and fourth objective functions become (8) and (9), respectively:

$$\text{minimize } Z_3 = \sum_{i=1}^{n_t} \widetilde{Q}_i^t, \tag{8}$$

$$\text{minimize } Z_4 = \sum_{i=1}^{n_t} \widetilde{R}_i^t. \tag{9}$$

2.3.2. Constraints. Constraints are related with service type, activity precedence, and service availability. Services of the required type can be assigned to an activity; therefore,

$$\sum_{v=1}^{S_k}\sum_{\tau\geq t}^{T} x_{i,j,k,v}^{t,\tau} = 1, \quad \forall \text{Type}\left(A_{i,j}^t\right) = k \ (k = 1, 2, \ldots, l). \quad (10)$$

Another constraint is the activity precedence. A service cannot be allocated to an activity with a preceding activity that has not been completed, which is expressed as follows:

$$\tau_0 \leq \tau_1, \quad \text{if } x_{i,j_0,k_0,v_0}^{t,\tau_0} \times x_{i,j_1,k_1,v_1}^{t,\tau_1} = 1, \text{and } A_{i,j_0}^t \in P_{i,j_1}^t, \quad (11)$$

where τ_0 is the assignment time for a preceding activity with assignment time τ_1. $x_{i,j_0,k_0,v_0}^{t,\tau_0} \times x_{i,j_1,k_1,v_1}^{t,\tau_1} = 1$ implies that S_{k_0,v_0} and S_{k_1,v_1} are assigned to A_{i,j_0}^t and A_{i,j_1}^t at τ_0 and τ_1, respectively.

The next constraint is service availability. A service cannot be allocated to other activities until it is released:

$$\sum_{w=0}^{p_{k,v}}\sum_{i,j} x_{i,j,k,v}^{t,\tau-w} \leq 1, \quad \forall \tau, T_i^t, j, k, v. \quad (12)$$

Constraint (12) can also be expressed by means of indicator variables as follows:

$$I\left(S_{k,v}^{\tau}\right) \geq x_{i,j,k,v}^{t,\tau}, \quad \forall \tau, T_i^t, j, k, v. \quad (13)$$

In other words, $x_{i,j,k,v}^{t,\tau} = 0$ if $I(S_{k,v}^{\tau}) = 0$, and $x_{i,j,k,v}^{t,\tau} = 1$ otherwise. The indicator $I(S_{k,v}^t)$ can prevent the system from allocating an unavailable service and resolve an unexpected situation in which a service becomes unavailable suddenly. For example, a service $S_{k,v}$ that is supposed to be allocated to a task at time $\tau \geq t$ may suddenly become unavailable because of a service-provider issue (e.g., contract expiration). The proposed GA can resolve this by setting $I(S_{k,v}^{\tau}) = 0$ for $\tau \geq t$ and then continuing to search the solutions.

Finally, because every decision variable is binary, we need the following:

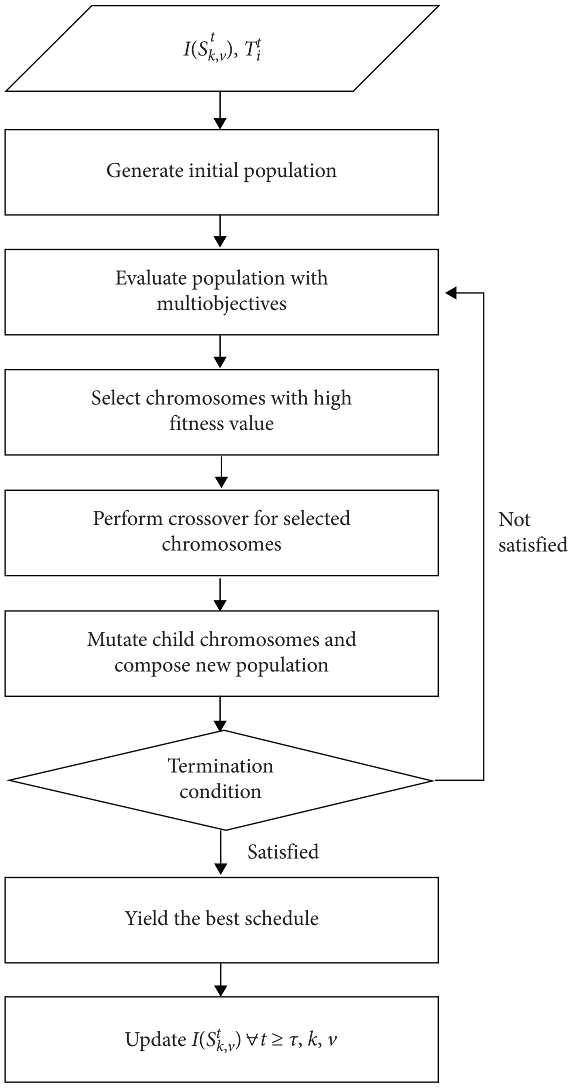
$$x_{i,j,k,v}^{t,\tau} \in \{0, 1\}, \quad \forall \tau, T_i^t, k, v, j. \quad (14)$$

3. Genetic Algorithm

This section describes the design procedure of the multiobjective GA to solve the mathematical model developed in the previous section. Figure 2 shows an illustrative flowchart of the proposed multiobjective GA, with which the schedule for a task T_i^t is constructed in the order of the registered time. That is, the proposed GA selects the proper services for a task, which are scheduled to process the task and then service availability is updated. Therefore, a task can be scheduled as soon as it is uploaded (that is, task can be scheduled in a real-time manner).

3.1. Chromosome Representation. The hth chromosome in the gth population is represented as $\chi^t[g][h] = (\chi_1^t[g][h], \chi_2^t[g][h], \ldots, \chi_{n_t}^t[g][h])$, where $\chi_i^t[g][h]$ is a chromosome for scheduling T_i^t. $\chi_i^t[g][h]$ consists of $\chi_{i,1}^t[g][h], \chi_{i,2}^t[g][h], \ldots, \chi_{i,m_i^t}^t[g][h]$, and $\chi_{i,1}^t[g][h]$ has two elements such as

$$\chi_{i,j}^t[g][h] = \left(v_{i,j}^t[g][h], \tau_{i,j}^t[g][h]\right), \quad (15)$$



$I(S_{k,v}^t)$, T_i^t
Generate initial population
Evaluate population with multiobjectives
Select chromosomes with high fitness value
Perform crossover for selected chromosomes
Mutate child chromosomes and compose new population
Termination condition
Not satisfied
Satisfied
Yield the best schedule
Update $I(S_{k,v}^t)\ \forall t \geq \tau, k, v$

Figure 2: Flowchart of the proposed approach.

where $v_{i,j}^t[g][h]$ is a service assigned to the activity and $\tau_{i,j}^t[g][h] = (\tau\tau_{i,j}^{t,s}[g][h], \tau_{i,j}^{t,s}[g][h]+1, \ldots, \tau_{i,j}^{t,e}[g][h])$ is a processing period of $A_{i,j}^t$ by $v_{i,j}^t[g][h]$. Here, $\tau_{i,j}^{t,s}[g][h]$ is the starting time and $\tau_{i,j}^{t,e}[g][h] = \tau_{i,j}^{t,s}[g][h] + p_{k,v} - 1$ is the completion time of $A_{i,j}^t$.

3.2. Generation of Initial Solution. If the GA generates initial solutions randomly, then convergence would take an extended amount of time because most solutions are infeasible. We therefore propose a method to generate initial solutions, $\chi^t[1][h]$, that are feasible, through Algorithm 1.

This algorithm generates solutions satisfying the constraints in Section 3.3 and operates ps times to generate $\chi^t[1][1], \chi^t[1][2], \ldots, \chi^t[1][ps]$, where ps is the number of chromosomes in a population.

3.3. Fitness Function. Pareto ranking is employed as a fitness function to select ps chromosomes and to yield the best chromosome in each generation. This method calculates a fitness value $s^t[g][h]$ of $\chi^t[g][h]$ as follows [28]:

```
Input: $I(S^{\tau}_{k,v})$ for all $\tau \geq t$, $k$, $v$
Procedure:
    Step 1. $i = 1$.
    Step 2. $j = 1$, and save the service type for $A^t_{i,j}$ as $k$, and randomly select $v^t_{i,j}$ and $\tau^{t,s}_{i,j}$ which satisfies:
        (1) $I(S^{\tau}_{k,v}) = 1$ for $\tau = \tau^s_{i,j}, \tau^s_{i,j}+1, \ldots, \tau^s_{i,j}+p_{k,v}-1$, and
        (2) $\tau^s_{i,j}$ is one of minimums among $\tau$ satisfying (1), and update $I(S^{\tau}_{k,v}) = 0$ for $v = v^t_{i,j}$ and
            $\tau = \tau^s_{i,j}, \tau^s_{i,j}+1, \ldots, \tau^s_{i,j}+p_{k,v}-1$.
    Step 3. Increase $j$ by 1 and save the service type for $A^t_{i,j}$ as $k$.
    Step 4. Randomly select $v^t_{i,j}$ and $\tau^s_{i,j}$ which satisfies:
        (1) $I(S^{\tau}_{k,v}) = 1$ for $\tau = \tau^s_{i,j}, \tau^s_{i,j}+1, \ldots, \tau^s_{i,j}+p_{i,j}-1$,
        (2) $\tau^s_{i,j} \geq \tau^s_{i,j'} + p_{i,j'} - 1$ for $A^t_{i,j'} \in PA^t_{i,j}$,
        (3) $\tau^s_{i,j}$ is one of minimums among $\tau$ satisfying both (1) and (2), and update $I(S^{\tau}_{k,v}) = 0$ for $v = v^t_{i,j}$ and
            $\tau = \tau^s_{i,j}, \tau^s_{i,j}+1, \ldots, \tau^s_{i,j}+p_{k,v}-1$.
    Step 5. $\chi^t_{i,j}[1][h] = (v^t_{i,j}, (\tau^s_{i,j}, \tau^s_{i,j}+1, \ldots, \tau^s_{i,j}+p_{k,v}-1))$.
    Step 6. If $i = n_t$ and $j = m^t_i$, terminate this algorithm. If $j = m^t_i$ and $i \neq n_t$, increase $i$ by 1
        and go to Step 2. If $j \neq m^t_i$, increase $j$ by 1 and go to step 3.
output: $\chi^t[1][h]$.
```

Algorithm 1: Generation of initial solutions of $\chi^t[1][h]$.

$$s^t[g][h] = \frac{1}{\mu\left(\gamma_1\left(\chi^t[g][h]\right), \gamma_2\left(\chi^t[g][h]\right), \gamma_3\left(\chi^t[g][h]\right), \gamma_4\left(\chi^t[g][h]\right)\right)}, \tag{16}$$

where $\gamma_i(\chi^t[g][h])\,(i = 1, 2, 3, 4)$ is the ranking of the value, Z_i, of the objective function with regard to $\chi^t[g][h]$ among chromosomes in the same generation and $\mu(\cdot)$ denotes a mean function. Table 3 demonstrates how to calculate the fitness value by means of Pareto ranking.

For example, because the Z_1 values of $\chi^t[g][i]\,(i = 1, 2, 3, 4)$ are 10, 12, 5, and 8, their ranks by $\text{rank}_1(\chi^t[g][i])\,(i = 1, 2, 3, 4)$ are 3, 4, 1, and 2, respectively (see the first column of Table 3, circled by a bold blue rectangle). The rankings of Z_1, Z_2, Z_3, and Z_4 with regard to $\chi^t[g][1]$ are 3, 1, 3, and 2, respectively, and the fitness value is calculated as $(1/\mu(3, 1, 3, 2)) = 0.444$ (see the first row of Table 3, delineated by a dashed red rectangle). Note that the solution with a smaller objective function value takes a higher ranking because they are to be minimized.

3.4. Genetic Operators. We adopt a uniform crossover operator that selects one of the crossover lines randomly and assigns chromosomes of parents to the children based on those lines [29]. In this paper, we avoided generating infeasible solutions as presented in Algorithm 2.

4. Experiment

In this section, we conduct experiments to verify that the proposed approach is practical under the experiment environment:

OS: Microsoft Windows 10 Home (×64)

CPU: Intel® Core™ i7-6700 CPU

RAM: 16.0 GB

Language: Python 3.6.9

First, a small toy problem is introduced to find all possible solutions and calculate their fitness values by means of a Pareto ranking method in Section 4.1, followed by checking the ranks of the solutions obtained by the proposed approach. Second, we apply the proposed approach to solve a more realistic, large-scale scheduling problem and examine the search time to establish a schedule in Section 4.2. The first experiment shows that our approach can find a schedule that is effective and sufficiently close to the optimal schedule. The second experiment is to verify the given schedule as efficient and appropriate for a large-scale practical problem in real time.

4.1. Effectiveness Verification. Figure 3 shows the task type introduced to compare the proposed method with an exhaustive search method. "S" and "H" are activities that require software and hardware services, respectively.

We assume three tasks, three software services, and three hardware services in this problem. Information about the tasks, such as maximum allowed due date (D^t_i), cost (C^t_i), minimum quality (Q^t_i), and reliability (R^t_i), is summarized in Table 4.

Table 5 supplies information on services such as cost $(c_{k,v})$, quality $(q_{k,v})$, reliability $(r_{k,v})$, and processing time $(p_{k,v})$.

Table 3: An example of the Pareto ranking method.

	Objective function value () = ranking of solutions on each objective function					Fitness value
	Z_1	Z_2	Z_3	Z_4		
$\chi^t[g][1]$	10(3)	6(1)	8(3)	16(2)		0.444
$\chi^t[g][2]$	12(4)	10(3)	6(2)	20(4)	$\Rightarrow$	0.308
$\chi^t[g][3]$	5(1)	8(2)	4(1)	15(3)		0.571
$\chi^t[g][4]$	8(2)	15(4)	10(4)	10(1)		0.364

```
Input: parent chromosomes $\chi^t[g][h_1]$ and $\chi^t[g][h_2]$ $(h_1 \neq h_2)$
Procedure:
    Step 1. Initialize $i$ as 1.
    Step 2. Initialize $j$ as 1.
    Step 3. Randomly select an element from $\{h_1, h_2\}$, and let $h$ be the chosen and $\tilde{h}$ be the other one.
    Step 4. $\chi^t_{i,j}[g+1][h_3] = \chi^t_{i,j}[g+1][h]$ and $\chi^t_{i,j}[g+1][h_4] = \chi^t_{i,j}[g+1][\tilde{h}]$.
    Step 5. If $\tau^{t,e}_{i,j}[g][h] \leq \min_{\bar{j}}(\tau^{t,s}_{i,\bar{j}}[g][\tilde{h}])$, $\tau^{t,e}_{i,j}[g][\tilde{h}] \leq \min_{\bar{j}}(\tau^{t,s}_{i,\bar{j}}[g][h])$, $\max_{\tilde{j}}(\tau^{t,e}_{i,\tilde{j}}[g][h]) \leq \tau^{t,s}_{i,j}[g][\tilde{h}]$, and
        $\max_{\tilde{j}}(\tau^{t,e}_{i,\tilde{j}}[g][h]) \leq \tau^{t,s}_{i,j}[g][\tilde{h}]$, then exchange $h$ and $\tilde{h}$ with a probability $\lambda$, where $A^t_{i,\tilde{j}} \in PA^t_{i,j}$ and $A^t_{i,\bar{j}} \in FA^t_{i,j}$.
        Otherwise, do nothing.
    Step 6. If $j = m^t_i$ and $i = n_t$, terminate. If $j = m^t_i$ and $i \neq n_t$, increase $i$ by 1 and go to Step 2. If $j \neq m^t_i$, increase $j$ by 1 and go to
        Step 3.
Output: children chromosomes $\chi^t[g+1][h_3]$ and $\chi^t[g+1][h_4]$.
```

Algorithm 2: Procedure of uniform crossover.

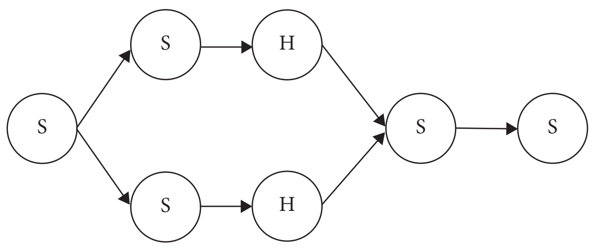


Figure 3: Task type for effectiveness verification.

GA parameters in the proposed approach are set as follows: the number of generations is 50; the number of solutions per each generation is 20. Therefore, the GA searches $50 \times 20 = 1000$ solutions. We calculate the ranks of all solutions, including the one obtained by the proposed approach. Any inferior solutions (e.g., those waiting to allocate a service even though it is available, solutions that allocate no service at all, and solutions that allocate a single service to every activity) are ignored when searching all possible solutions. Table 6 summarizes all possible solutions, which are listed in the descending order of fitness values. The values in bold indicate the solution obtained by the proposed GA.

As seen in Table 6, our approach has found the 172th solution among 34,012,224 solutions, which lies in the top 0.00051%. In other words, the proposed algorithm can find a solution which is very close to the optimal. Because the proposed algorithm makes a schedule after a task is uploaded in the task pool, we expect that the number of tasks has little effect on the result. From the experiment result and the real-time property of the algorithm, we can conclude that the proposed algorithm is effective for task scheduling in cloud manufacturing.

Table 4: Task information.

Task	D^t_i	C^t_i	Q^t_i	R^t_i
T^1_1	10	160	98.5	0.98
T^1_2	8	170	99.0	0.99
T^2_1	14	150	98.0	0.98

4.2. Efficiency Verification. Next, a CM platform is assumed dealing with a more complicated and realistic task, depicted in Figure 4, which was adopted from [30].

We randomly generate six situations where 5, 10, 15, 20, 25, and 30 tasks are uploaded at a day, and we calculate search time to establish the schedules in each situation. We assume that every task is uploaded at t, and D^t_i, C^t_i, Q^t_i, and R^t_i are uniformly distributed in the intervals [30, 60], [400, 450], [95, 100], and [0.95, 1.00], respectively.

A total of 30 software services and 15 hardware services are virtualized and shared in the platform, and every service is idle at first. We assume that $q_{k,v}$, $r_{k,v}$, and $p_{k,v}$ follow uniform distributions in the intervals [95, 100], [0.95, 1.00], and [1, 3], respectively. The costs of hardware and software services are assumed to be different from each other; that is, $c_{k,v}$ of software and hardware service is

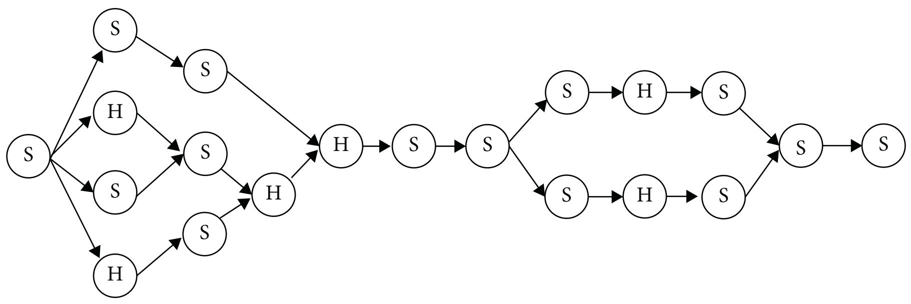


Figure 4: Task type for efficiency verification.

Table 5: Service information.

Software service	$c_{k,v}$	$q_{k,v}$	$r_{k,v}$	$p_{k,v}$
$S_{1,1}$	22	100	1.00	1
$S_{1,2}$	18	99	0.98	2
$S_{1,3}$	14	97	0.97	3
Hardware service				
$S_{2,1}$	40	100	1.00	1
$S_{2,2}$	34	98	0.98	2
$S_{2,3}$	28	97	0.97	3

Table 6: Objective functions, ranks, and fitness values of all possible solutions.

Rank	Value				Rank				Fitness value
	Z_1	Z_2	Z_3	Z_4	Z_1	Z_2	Z_3	Z_4	
1	1	94	0.0000	0.3187	353,454	5,661,611	1	3,899,820	4.034×10^{-7}
2	1	94	0.0000	0.3187	353,454	5,661,611	1	3,899,547	4.034×10^{-7}
3	1	96	0.0000	0.3101	353,454	6,921,456	1	2,932,620	3.918×10^{-7}
4	1	96	0.0000	0.3101	353,454	6,921,456	1	2,934,610	3.917×10^{-7}
5	1	90	0.0000	0.3361	353,454	4,163,149	1	6,170,497	3.742×10^{-7}
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
172	**1**	**96**	**0.1071**	**0.3195**	**272,978**	**6,921,456**	**9,569,507**	**4,248,174**	1.904×10^{-7}
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
34,012,224	7	98	0.1428	0.3727	14,330,889	7,321,564	14,768,747	11,846,972	8.287×10^{-8}

Table 7: Search time according to the numbers of solutions and tasks (unit: seconds).

		The number of solutions to be searched			
		1,000	5,000	10,000	Average scheduling time for each task
The number of tasks to be scheduled	5	140	767	1,551	310
	10	225	1,646	2,050	205
	15	686	1,939	6,442	429
	20	831	4,267	7,964	398
	25	1,068	7,694	10,141	406
	30	1,322	9,012	12,438	415

assumed to follow uniform distributions in [28, 40] and [95, 100], respectively.

Table 7 shows the search times when the numbers of solutions are 1000, 5000, and 10,000 and tasks are 5, 10, 15, 20, 25, and 30. It is obvious that as more solutions are searched it takes longer but is more likely to find the best solution. The rightmost column is the average scheduling time per task when 10,000 solutions are considered.

As seen, the maximum search time is 12,438 seconds (3.455 hours) for 10,000 solutions and 30 tasks. A search time of three and a half hours is a relatively long time, raising the concern that the proposed approach is inadequate to be applied to in real time. However, the average time to schedule a task when we searched 10,000 solutions is 205~420 seconds, approximately 3~7 minutes. In practice, it is not usual for a complicated task to be uploaded every 7 minutes. In addition, the considered task is overly complicated comparing to realistic tasks, implying that the average time will be much lower than the experiment result provided in this study. Thus, once a complete and full schedule has been established, which may take several hours, a couple of tasks can be added to the existing schedule, which can be updated within 10 minutes, to produce an efficient schedule. Therefore, we conclude that the proposed approach is efficient and can be used in practice with a large number of complicated tasks and services. Even though the proposed algorithm is applied to establish schedules under an unreal situation in terms of the number of tasks and services, it can be applied to real tasks and services uploaded in CM such as in [8].

5. Conclusion

CM is a manufacturing model based on the concept of sharing manufacturing resources among manufacturing enterprises to deal with highly customized requests of customers. In CM, each request is regarded a task, and it is assigned to one or more enterprises, called task scheduling. The objective of the task scheduling is to maximize customer satisfaction, and therefore, tardiness, cost, quality, and so forth should be considered. In addition, the task should be assigned as soon as possible for practical usage.

This paper addressed a real-time and multiobjective task-scheduling problem to minimize total tardiness, cost, quality, and reliability penalties in CM. This problem was formalized as a binary integer programming model, and the MOGA was designed to solve the model by focusing on generating feasible solutions. From the experiment, we showed that the schedule based on the proposed approach is near the optimal schedule for small problems. In addition, we verified that the proposed approach establishes effective schedules in a real-time manner for a more realistic large-scale problem.

In future research, we will develop a GA that does not generate infeasible or inferior solutions. This will surely reduce the search time remarkably. In addition, we will conduct practical research to apply our algorithm to the commercialized cloud manufacturing system. Finally, we will improve the proposed multiobjective GA to apply it to train the deep learning model-based CM scheduler.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea Government (MSIT) (2019R1A2C1088255).

References

[1] G. Ahn, Y.-J. Park, and S. Hur, "The dynamic enterprise network composition algorithm for efficient operation in cloud manufacturing," *Sustainability*, vol. 8, no. 12, p. 1239, 2016.

[2] G. Ahn, Y.-J. Park, and S. Hur, "Probabilistic graphical framework for estimating collaboration levels in cloud manufacturing," *Sustainability*, vol. 9, no. 2, p. 277, 2017.

[3] D. Wu, M. J. Greer, D. W. Rosen, and D. Schaefer, "Cloud manufacturing: strategic vision and state-of-the-art," *Journal of Manufacturing Systems*, vol. 32, no. 4, pp. 564–579, 2013.

[4] B. Huang, C. Li, C. Yin, and X. Zhao, "Cloud manufacturing service platform for small-and medium-sized enterprises," *The International Journal of Advanced Manufacturing Technology*, vol. 65, no. 9–12, pp. 1261–1272, 2013.

[5] G. Ahn, Y.-J. Park, and S. Hur, "Performance computation methods for composition of tasks with multiple patterns in cloud manufacturing," *International Journal of Production Research*, vol. 57, no. 2, pp. 517–530, 2019.

[6] L. Zhou, L. Zhang, C. Zhao, Y. Laili, and L. Xu, "Diverse task scheduling for individualized requirements in cloud manufacturing," *Enterprise Information Systems*, vol. 12, no. 3, pp. 1–19, 2018.

[7] Y. Cao, S. Wang, L. Kang, and Y. Gao, "A TQCS-based service selection and scheduling strategy in cloud manufacturing," *The International Journal of Advanced Manufacturing Technology*, vol. 82, no. 1–4, pp. 235–251, 2016.

[8] Y. Liu, X. Xu, L. Zhang, L. Wang, and R. Y. Zhong, "Workload-based multi-task scheduling in cloud manufacturing," *Robotics and Computer-Integrated Manufacturing*, vol. 45, pp. 3–20, 2017.

[9] H. Jiang, J. Yi, S. Chen, and X. Zhu, "A multi-objective algorithm for task scheduling and resource allocation in cloud-based disassembly," *Journal of Manufacturing Systems*, vol. 41, pp. 239–255, 2016.

[10] W. Li, C. Zhu, L. T. Yang, L. Shu, E. C. H. Ngai, and Y. Ma, "Subtask scheduling for distributed robots in cloud manufacturing," *IEEE Systems Journal*, vol. 11, no. 2, pp. 941–950, 2017.

[11] N. D. Hoang, "Image processing-based pitting corrosion detection using metaheuristic optimized multilevel image thresholding and machine-learning approaches," *Mathematical Problems in Engineering*, vol. 2020, Article ID 6765274, 19 pages, 2020.

[12] W. Chen, Z. Li, and J. Guo, "A VNS-EDA algorithm-based feature selection for credit risk classification," *Mathematical Problems in Engineering*, vol. 2020, Article ID 4515480, 14 pages, 2020.

[13] X. Xue, J. Lu, and J. Chen, "Using NSGA-III for optimising biomedical ontology alignment," *CAAI Transactions on Intelligence Technology*, vol. 4, no. 3, pp. 135–141, 2019.

[14] H. S. Pannu, D. Singh, and A. K. Malhi, "Improved particle swarm optimization based adaptive neuro-fuzzy inference system for benzene detection," *CLEAN–Soil, Air, Water*, vol. 46, no. 5, Article ID 1700162, 2018.

[15] L. R. Rodrigues and J. P. P. Gomes, "TLBO with variable weights applied to shop scheduling problems," *CAAI Transactions on Intelligence Technology*, vol. 4, no. 3, pp. 148–158, 2019.

[16] T. Garai and H. Garg, "Multi-objective linear fractional inventory model with possibility and necessity constraints under generalised intuitionistic fuzzy set environment," *CAAI Transactions on Intelligence Technology*, vol. 4, no. 3, pp. 175–181, 2019.

[17] S. Lalwani, H. Sharma, A. Verma, and R. Kumar, "Efficient discrete firefly algorithm for Ctrie based caching of multiple sequence alignment on optimally scheduled parallel

machines," *CAAI Transactions on Intelligence Technology*, vol. 4, no. 2, pp. 92–100, 2019.

[18] S. Jiang, C. Zhang, and S. Chen, "Sequential hybrid particle swarm optimization and gravitational search algorithm with dependent random coefficients," *Mathematical Problems in Engineering*, vol. 2020, Article ID 1957812, 17 pages, 2020.

[19] Y. Peng, K. Lei, X. Yang, and J. Peng, "Improved chaotic quantum-behaved particle swarm optimization algorithm for fuzzy neural network and its application," *Mathematical Problems in Engineering*, vol. 2020, Article ID 9464593, 11 pages, 2020.

[20] R. L. Kadri and F. F. Boctor, "An efficient genetic algorithm to solve the resource-constrained project scheduling problem with transfer times: the single mode case," *European Journal of Operational Research*, vol. 265, no. 2, pp. 454–462, 2018.

[21] F. Lu, H. Bi, M. Huang, and S. Duan, "Simulated annealing genetic algorithm based schedule risk management of IT outsourcing project," *Mathematical Problems in Engineering*, vol. 2017, Article ID 6916575, 17 pages, 2017.

[22] P.-H. Lu, M.-C. Wu, H. Tan, Y.-H. Peng, and C.-F. Chen, "A genetic algorithm embedded with a concise chromosome representation for distributed and flexible job-shop scheduling problems," *Journal of Intelligent Manufacturing*, vol. 29, no. 1, pp. 19–34, 2018.

[23] Y.-B. Woo, S. Jung, and B. S. Kim, "A rule-based genetic algorithm with an improvement heuristic for unrelated parallel machine scheduling problem with time-dependent deterioration and multiple rate-modifying activities," *Computers & Industrial Engineering*, vol. 109, pp. 179–190, 2017.

[24] C. Chamnanlor, K. Sethanan, C.-F. Chien, and M. Gen, "Re-entrant flow shop scheduling problem with time windows using hybrid genetic algorithm based on auto-tuning strategy," *International Journal of Production Research*, vol. 52, no. 9, pp. 2612–2629, 2014.

[25] R. C. Chen, J. Chen, T. S. Chen, C. C. Huang, and L. C. Chen, "Synergy of genetic algorithm with extensive neighborhood search for the permutation flowshop scheduling problem," *Mathematical Problems in Engineering*, vol. 2017, Article ID 3630869, 9 pages, 2017.

[26] L. Tang, L. Zheng, H. Cao, and N. Huang, "An improved multi-objective genetic algorithm for heterogeneous coverage RFID network planning," *International Journal of Production Research*, vol. 54, no. 8, pp. 2227–2240, 2016.

[27] Y. Zhang, X. Hu, and C. Wu, "A modified multi-objective genetic algorithm for two-sided assembly line re-balancing problem of a shovel loader," *International Journal of Production Research*, vol. 56, no. 9, pp. 3043–3063, 2017.

[28] S. Kukkonen and J. Lampinen, "Ranking-dominance and many-objective optimization," in *Proceedings of IEEE Congress on Evolutionary Computation*, pp. 3983–3990, Singapore, September 2007.

[29] G. Syswerda, "Uniform crossover in genetic algorithms," in *Proceedings of the 3rd international Conference on Genetic Algorithms*, pp. 2–9, San Francisco, CA, USA, June 1989.

[30] F. Tao, Y. LaiLi, L. Xu, and L. Zhang, "FC-PACO-RM: a parallel method for service composition optimal-selection in cloud manufacturing system," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 4, pp. 2023–2033, 2013.