Computer Modeling in
Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2021.015314

ARTICLE

A Homogeneous Cloud Task Distribution Method Based on an Improved Leapfrog Algorithm

Yunliang Huo[1], **Ji Xiong**[1,*], **Zhixing Guo**[1], **Qianbing You**[1] **and Yi Peng**[2]

[1]School of Mechanical Engineering, Sichuan University, Chengdu, 610065, China

[2]Chengdu Yigao Intelligent Technology Co., Ltd., Chengdu, 610065, China

[*]Corresponding Author: Ji Xiong. Email: 13668149296@163.com

Received: 08 December 2020 Accepted: 08 April 2021

ABSTRACT

Cloud manufacturing is a new manufacturing model with crowd-sourcing characteristics, where a cloud alliance composed of multiple enterprises, completes tasks that a single enterprise cannot accomplish by itself. However, compared with heterogeneous cloud tasks, there are relatively few studies on cloud alliance formation for homogeneous tasks. To bridge this gap, a novel method is presented in this paper. First, a homogeneous cloud task distribution model under cloud environment was constructed, where services description, selection and combination were modeled. An improved leapfrog algorithm for cloud task distribution (ILA-CTD) was designed to solve the proposed model. Different from the current alternatives, the initialization operator and the leapfrog operator in ILA-CTD can ensure that the algorithm always searches the optimal solution in the feasible space. Finally, the processing of task allocation for 1000 pieces of medical labeling machine bottom plates was studied as a case to show the feasibility of the proposed method. The superiority of ILA-CTD was also proven based on more optimal solutions found, compared with the three other methods.

KEYWORDS

Cloud manufacturing; service composition; tasks distribution; intelligent optimization; leapfrog algorithm

1 Introduction

The application of modern technologies (the Internet of Things [1,2], service-oriented technology [3], cloud computing [4], big data [5], etc.) had a profound impact on manufacturing methods. Cloud manufacturing (CMfg) is a network-based and knowledge-enhanced manufacturing model, it is a specific form of the service-oriented manufacturing paradigm [6]. Generally, there are two types of clouds: private clouds in large enterprises [6,7] and public clouds for small and medium enterprises (SMEs) [8]. The application of CMfg in SMEs would better reflect the characteristics of CMfg, such as centralized management of distributed resources, crowd-sourcing manufacturing, and highly shared manufacturing resources [9].

There is consistent requirement in CMfg in SMEs, namely that a large order needs to be fulfilled in a short time, this is undertaken by an alliance of multiple SMEs [10]. A single SME is unable to complete such a large order in a timely manner. A reasonable distribution of a large order to multiple SMEs can drastically reduce the time required to fulfill the order, because the order can be executed parallelly. The global optimization for this activity can be achieved through reasonable task allocation according to the capabilities of the alliance members. Two crucial steps are needed to form a cloud alliance: selection of members (services selection) and reasonable task distribution (services composition) [11].

Selection of alliance members, namely services selection (SS) can be achieved through methods such as semantic similarity [12], QoS (quality of services) [13] and rough-fuzzy approach [14]. For task distribution, cloud manufacturing relates to two types: heterogeneous tasks that require different services [15], and homogeneous tasks that require the same services [16]. The difference is that the latter selects services based only on QoS, and even if the production capacity of an individual service is insufficient, a large number of available services will still be obtained. However, not every available service can be assigned a task because there is the constraint of starting quantity. Therefore, how to reasonably allocate mass homogeneous tasks to such services is a complicated question with many constraints such as production capacity, QoS, and starting quantity. To propose a solution to this problem, an improved leapfrog algorithm for cloud task distribution (ILA-CTD) is presented in this study. A novel initialization operator was designed in this method to avoid the solution modification caused by the restriction of the starting quantity. The solution obtained by the leap operator satisfies the restriction of the starting quantity. Pareto optimal theory was applied to leapfrog algorithm, so that the proposed algorithm has the ability to deal with multi-objective optimization problems.

This article is organized as follows: Section 2 reviews related works; Section 3 presents the proposed model; Section 4 introduces the novel task distribution method; Section 5 describes a case study with the results discussions; and Section 6 concludes the study with remarks and suggestions for future work.

2 Related Works

2.1 Service Selection (SS)

Manufacturing services are usually designed to be user-friendly (i.e., services can be easily identified through accurate description of QoS). After alternative services are found, discrimination of the services with overlapping or identical functionalities based on QoS can be achieved [17,18]. However, things become more complex when a task consists of several subtasks, that need multiple services to accomplish collaboratively [16].

Many methods have been developed to rank and select services. Zhao et al. [19] proposed an optimal service selection approach using crowd-based cooperative computing, whose main contribution was optimally balancing the QoS and the synergy effect. Eisa et al. [18] presented a Multi-Criteria Decision-Making model to rank services based on various QoS attributes; unlike other approaches, their work was based on a real cloud provider (Amazon). Hussain et al. [20] presented a novel customer-centric Methodology for Optimal Service Selection (MOSS) in a cloud environment. Bouzary et al. [21] used TF-IDF (term frequency-inverse document frequency) to identify services that satisfy QoS. A modified interval DEA model with undesirable outputs has been adopted to achieve more accurate web service selection [22]. In addition, the dynamic change of consumer requirements was also considered by Devi et al. [23], they proposed a Linear Programming model to rank and select services dynamically.

Obviously, QoS plays an important role in cloud services selection. To build upon previous research, a selection method for homogeneous services based on QoS is proposed in this work.

2.2 Task Distribution and Services Composition

From ants to human beings, animals have the ability to cooperate, communicate and divide labor among individuals, which is inspiring collaborative manufacturing. Chen et al. [16] proposed an improved multi-objective evolutionary algorithm based on the decomposition-particle swarm optimization (MOEA/D-PSO) to obtain the optimal combination of services. Aimed at minimizing the making span, monetary and energy costs of tasks in the cloud-fog paradigm, a two-tier bipartite graph task allocation approach was presented by Gad-Elrab et al. [24] based on fuzzy set theory. Gigliotta et al. [25] examined the issues in task allocation in homogeneous communicating robots using the evolutionary algorithm. Sharma et al. [26] proposed an improved cloud task allocation strategy using a modified K-means clustering technique. A hybrid approach combining the features of genetic algorithm and the analytical hierarchy process, was implemented by Mostafa et al. [27] to distribute tasks to service suppliers. A multi-objective genetic algorithm was used by Jiang et al. [28] to allocate the disassembly tasks in the cloud environment. Jatoth et al. [29] presented an Optimal Fitness Aware Cloud Service Composition (OFASC) to balance multiple parameters of QoS. Zhou et al. [30] used evolutionary algorithms for many-objective cloud services composition. Somasundaram et al. [31] designed and developed a cloud resources broker (CLOUDRB), which integrated CLOUDRB with deadline-based job scheduling. They also developed a particle swarm optimization (PSO)-based resource allocation mechanism, to allocate users' requirements to cloud resources in a near-optimal manner.

In summary, task distribution (services combination) is a complex problem, and intelligent evolution algorithms (such as GA, EA and PSO) have made substantial contribution to address the complexities. Inspired by the previous efforts, ILA-CTD is proposed in this study to address the homogeneous cloud task distribution problem.

3 Homogeneous Task Distribution Model

3.1 Problem Statement

The manufacturing resources in CMfg are encapsulated as manufacturing service in a cloud resources pool. When the manufacturing tasks published in CMfg, three stages follow, namely, task decomposition, services selection, and services composition. The homogeneous cloud task distribution model is shown in Fig. 1.

Manufacturing resources of factories are virtualized to various cloud services in a cloud service pool. When the tasks are uploaded, the service pool will be searched and available services will be selected. Then, the optimal service combination will be determined to identify an optimized cloud manufacturing alliance. The factories in the alliance will be notified to perform manufacturing tasks. The focus of our work is services selection and combination optimization, specifically for homogeneous tasks. This is expressed in Eq. (1):

$$\begin{cases} T^{p}_{(n)} = \{P_1, P_2, \ldots, P_i, \ldots, P_n\} \\ S^{c}_{(f)} = \left\{S_1, S_2, \ldots, S_j, \ldots, S_f\right\} \\ M_j \leq s^j \end{cases} \tag{1}$$

$T^{p}_{(n)}$ indicates cloud tasks with n products, P_j is the ith product; $S^{c}_{(f)}$ is the set of available services; S_j is the jth available service; s^j denotes the manufacturing capacity of S_j; and M_j is the starting quantity of S_j. Services selection can be achieved based on QoS, and then, the rest of the problem converts to the distribution of n products to f services with both QoS and cloud alliance quality considered.

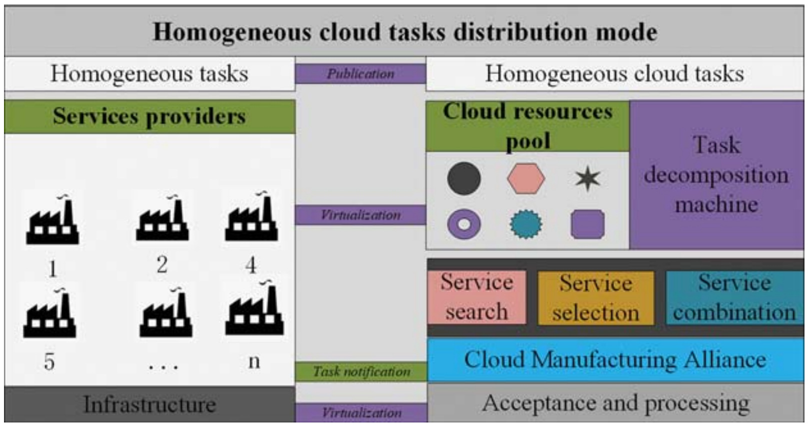

Figure 1: The model of homogeneous cloud task distribution

3.2 Service Description and Selection

3.2.1 Manufacturing Service Description in the Cloud

Offline services selection mainly focuses on QoS, which involves cost, time, and quality. However, for cloud services, the attributes that affect the quality of the cloud manufacturing alliance also need to be considered. Evaluation indexes proposed by Chen give a comprehensive description of cloud manufacturing services; cloud manufacturing services (C-QoS) were expressed as 6 tuples (C-$QoS = \{SI, G, C_0, T, C, Q\}$) [16].

1) SI shown in Eq. (2) is the quality consistency:

$$\begin{cases} SI_i = \dfrac{\sum_{j=1}^{f}\sum_{k=1}^{q}\left(Q^{jk}.Sgn\left(S^j\right)-\mu_k\right)^2}{k.\sum_{j=1}^{f}Sgn\left(S^j\right)} \\ \mu_k = \dfrac{\sum_{j=1}^{f}Q^{jk}.Sgn\left(S^j\right)}{\sum_{j=1}^{f}Sgn\left(S^j\right)} \end{cases} \tag{2}$$

SI_i is the quality consistency of ith cloud alliance; Q^{jk} is the evaluation value of kth quality index of S^j in its history; S^j is jth available service; $Sgn\left(S^j\right)$ is the status of service S^j. If S^j is selected, $Sgn(S^j) = 1$; else $Sgn(S^j) = 0$. f is the number of available services, and q is the total number of quality indexes.

2) G shown in Eq. (3) is the composability:

$$\begin{cases} G_i = \dfrac{\sum_{j=1}^{f} Sgn\left(S^j\right).G^j}{\sum_{j=1}^{f} Sgn\left(S^j\right)} \\ G^j = \dfrac{Go^j}{Gn^j} \end{cases} \tag{3}$$

G_i is the composability of the ith cloud alliance; Go^j is the number of times that S^j has been used in a combination in its history; Gn^j is the number of times that S^j has been used in its history.

3) C_0 shown in Eq. (4) is communication ability:

$$\begin{cases} Co_i = \dfrac{\sum_{j=1}^{f} Sgn\left(S^j\right).Co^j}{\sum_{j=1}^{f} Sgn\left(S^j\right)} \\ Co^j = \sum\limits_{h=1}^{p} Co^{jh}/p \end{cases} \tag{4}$$

Co_i is the communication ability of the ith cloud alliance; Co^{jh} is the evaluation value of communication ability given by the hth service provider who has cooperated with S^j; p is the number of services that have collaborated with S^j.

4) T shown in Eq. (5) is the time consumed by service:

$$\begin{cases} T_i = \max\left(Tm^j + Tt^j\right) \\ Tm^j = s_i{}^j.Tma^j \\ Tt^j = Tta^j \end{cases} \tag{5}$$

T_i is the consumed time of the ith cloud alliance; Tma^j is the time consumed by S^j for producing a unit product; Tta^j is the time consumed when the unit product is transported by S^j; and s^j is the number of products undertaken by S^j.

5) C shown in Eq. (6) is the cost:

$$\begin{cases} C_i = \sum\limits_{j=1}^{f} \left(Cm^j + Ct^j\right) \\ Cm^j = s^j.Cma^j \\ Ct^j = s^j.Cta^j \end{cases} \tag{6}$$

C_i is the cost of the ith cloud alliance; Cma^j is the cost consumed by S^j for producing a unit product; Cta^j is the cost consumed by S^j when the unit product is transported.

6) Q shown in Eq. (7) is the quality:

$$\begin{cases} Q_i = \dfrac{\sum_{j=1}^{f} Sgn\left(S^j\right).Q^j}{\sum_{j=1}^{f} Sgn\left(S^j\right)} \\ Q^j = \sum\limits_{k=1}^{q} Q^{jk}/n_Q \end{cases} \tag{7}$$

Q^{jk} is the evaluation value of the kth quality index of S^j in its history; and n_Q is the total number of evaluations of Q^{jk} in its history.

3.2.2 Services Selection

For r candidate services in the cloud pool, incapable services should first be eliminated from selection. Then, f $(f < r)$ available services will exist. As shown in Eq. (8), a selection method based on QoS was designed to filter incapable services; the task requirements are expressed as $R = [\mathrm{r}_1, r_2, \ldots, r_k, \ldots, r_c]$, and the attributes of jth service are expressed as $S_j^a = [a_1, a_2, \ldots, a_k, \ldots, a_c]$.

$$\begin{cases} E_j = [e_1, e_2, \ldots, e_k, \ldots, e_c] \\ e_k = r_k - a_k \\ \forall\, e_k \in E_j \\ if\;\; e_k < 0; \quad S_j \;is\; incapable \\ else; \quad S_j \;is\; capable \end{cases} \tag{8}$$

r_k is the kth requirement, and a_k is the value of the service attribute corresponding to r_k.

3.3 Distribution Model

The distribution model of n products to f available services is shown in Eqs. (9)–(11). As $s^j \geq M_j$ must be satisfied, a situation may arise, namely, that not every available service can be assigned with tasks if n is relatively small or f is relatively large.

$$M = \left[\begin{array}{ccccccccccc|cc} S_1 & Tm^1 & Tt^1 & Cm^1 & Ct^1 & Q^{11} & \cdots & Q^{1q} & Gn^1 & Go^1 & Co^1 & M_1 & s^1 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ S_j & Tm^j & Tt^j & Cm^j & Ct^j & Q^{j1} & \cdots & Q^{jq} & Gn^j & Go^j & Co^j & M_j & s^j \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ S_f & Tm^f & Tt^f & Cm^f & Ct^f & Q^{f1} & \cdots & Q^{fq} & Gn^f & Go^f & Co^f & M_f & s^f \end{array}\right] \tag{9}$$

Therefore, a complex problem arises, because product quantities undertaken by each service and different service combinations will constitute different cloud alliance options. We assign a reasonable quantity and s^j (shown in Eq. (9)) and optimize objectives (shown in Eq. (10)) under constraints (shown in Eq. (11)):

$$F = Max\,(Q, G, C_0)\,\&\&Min\,(SI, T, C) \tag{10}$$

$$\sum_{j=1}^{f} s^j = n$$
$$s^j \geq M^j \quad \text{or} \quad s^j = 0 \tag{11}$$
$$T_{\mathrm{i}} \leq T_r$$
$$C_i \leq C_r$$

T_r and C_r represent consumer expectations of time and cost, respectively.

4 Solution Methods

The leapfrog method is an efficient algorithm with both hereditary and group behavior [32–34]. However, the classic leapfrog algorithm is incapable of solving the above model because it is a multi-objective optimization problem, and many infeasible solutions will be generated because of the constraint of starting quality. To solve the presented model, an improved leapfrog algorithm for cloud task distribution (ILA-CTD) is proposed.

4.1 Basic Definitions

Pareto domination: Assuming that a and b are two solutions of a multi-objective function F with m benefit-oriented targets and n cost-oriented targets, if $\forall\, i$, $F_i(a) \leq F_i(b)$ $(i = 1, 2, \ldots, m)$ and $F_j(a) \geq F_j(b)$ $(j = 1, 2, \ldots, n)$, where F_i is the benefit-oriented target and F_j is the cost-oriented target, and at least one strict inequality holds, then b dominates a.

Non-dominated solution: Assuming that c is a feasible solution of a multi-objective function F with m targets, if there is no other solution d that satisfies $F_i(c) \leq F_i(d)$ $(i = 1, 2, \ldots, m)$ and $F_j(a) \geq F_j(b)$ $(j = 1, 2, \ldots, n)$, then c is a non-dominated solution.

Elite archives: Set of non-dominated solutions obtained in the process of searching for the best solution.

4.2 ILA-CTD Operators

4.2.1 Initialization of Frog Population

A solution of the model is mapped to a frog $U_i = (s_i^1, \ldots, s_i^j, \ldots, s_i^f)$ in ILA-CTD, and p frogs form a population P. Different from the classic leapfrog algorithm in which the initialization of the population is achieved via the rand methods, in this work, a novel initialization method shown in Eq. (12), was designed to satisfy the constraint in Eq. (13).

$$\begin{cases} j = unidrnd\,(f) \\ k = (a * unidrnd\,(b)) * \left(n - M^j\right) \\ if \;\; r \leq \left(c + M^j/T\right); \quad s^j = 0 \\ else \;\; s^j = M^j + unidrnd\,(k) \end{cases} \tag{12}$$

unidrnd(f) is a function that produces an integer from 0 to f; a and b are the initial speed factors; c is the diversity factor; and r is a random number between 0 and 1.

$$\sum_{j=1}^{f} s^j = n \tag{13}$$

n is the total number required to be distributed. Obviously, all solutions obtained through the designed initialization method satisfy the constraint of starting quality, and the diversity of population P is also guaranteed via factor c.

4.2.2 Fitness Function

A fitness function shown in Eq. (14) was designed to rank frogs, where f_i is the fitness value of the ith frog in a population.

$$f_i = \begin{cases} 1 & U_i \ is \ non-dominated \ solution \\ 0 & else \end{cases} \tag{14}$$

4.2.3 Grouping

Dividing p frogs into q groups, where $q < p$. The ith group is called *memeplex*(i). The detailed grouping rules are as follows: (1) sorting frogs by fitness value. (2) assigning the sorted frogs to each group in turn. For example, $q = 3$; *1th* frog enters *memeplex(1)*; *2th* frog enters *memeplex(2)*; *3th* frog enters *memeplex(3)*; and *4th* frog enters *memeplex(1)* again.

4.2.4 Leap Operator $\tau(U_i)$

The search process constantly makes the worst frog in each *memeplex* leap to a better position. The leap operator $\tau(U_b)$ is shown in Eq. (15):

$$\begin{aligned} &U_{\mathrm{w}} = \left(s_w{}^1, \ldots, s_w{}^j, \ldots, s_w{}^f\right) \\ &U_{\mathrm{b}} = \left(s_b{}^1, \ldots, s_b{}^j, \ldots, s_b{}^f\right) \\ &(U_{\mathrm{w}} = U_{\mathrm{w}} + U_{\mathrm{b}} - U_{\mathrm{w}}) \end{aligned} \tag{15}$$

U_w is the worst frog in a *memeplex*, and U_b is the best frog. It is easy to prove that the solution obtained via the leap operator also satisfies the constraint of starting quality, which is more efficient than the other methods that consume time to amend infeasible solutions.

Proof. If $U_i = (s_i^1, \ldots s_i^j, \ldots s_i^f)$ is a solution that satisfies the constraint of starting quality, then $s_i^j > M^j$ and $\sum_{j=1}^{f} s_i^j = n$. Assuming that $U_w = (s_w^1, \ldots, s_w^j, \ldots, s_w^f)$ and $U_b = (s_b^1, \ldots, s_b^j, \ldots, s_b^f)$ are two solutions that satisfy the constraint of starting quality, then Eq. (16) follows:

$$\begin{aligned} U_{\mathrm{b}} - U_{\mathrm{w}} &= \sum_{j=1}^{f} s_b^j - \sum_{j=1}^{f} s_w^j \\ &= 0 \end{aligned} \tag{16}$$

Hence, $U_w = U_w + (U_b - U_w)$ satisfies constraint that $\sum_{j=1}^{f} s_w^j = n$, for s_w^j of the new U_w, the Eq. (17) is always existence.

$$\begin{aligned} s_{nw}^j &= s_w^j + s_b^j - s_w^j \\ &= s_b^j \end{aligned} \tag{17}$$

U_b is feasible and satisfies the constraint of starting quality. Therefore, $s_b^j > M^j$ is also satisfied. Thus, the feasibility of the solution obtained using the leap operator is proved.

4.2.5 Local Search

Local search replaces the worst frog with a better one in each *memeplex*. The flowchart is shown in Fig. 2.

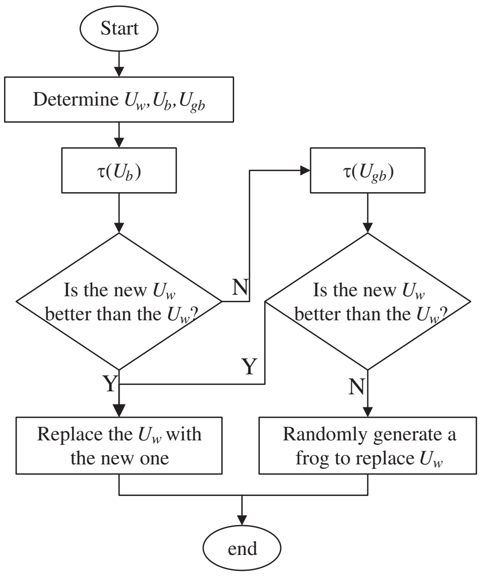
Start
Determine U_w, U_b, U_{gb}
$\tau(U_b)$
$\tau(U_{gb})$
Is the new U_w better than the U_w?
N
Y
Is the new U_w better than the U_w?
Y
N
Replace the U_w with the new one
Randomly generate a frog to replace U_w
end

Figure 2: Flowchart of local search

Step 1: Find the local worst frog U_w, the local best frog U_b, and the global best frog U_{gb}.

Step 2: Update U_w by using the leap operator $\tau(U_b)$.

Step 3: Is the new U_w better than the old one? If yes, go to Step 4; else, go to Step 5.

Step 4: Replace the U_w with the new one.

Step 5: Update the U_w by using the leap operator $\tau(U_{gb})$.

Step 6: Is the new U_w better than the existing U_w? If yes, go to Step 4; else, go to Step 7.

Step 7: Randomly generate a frog to replace U_w.

4.3 Process of ILA-CTD

Based on the above operators, the flowchart of ILA-CTD is shown in Fig. 3. The detailed steps are as follows:

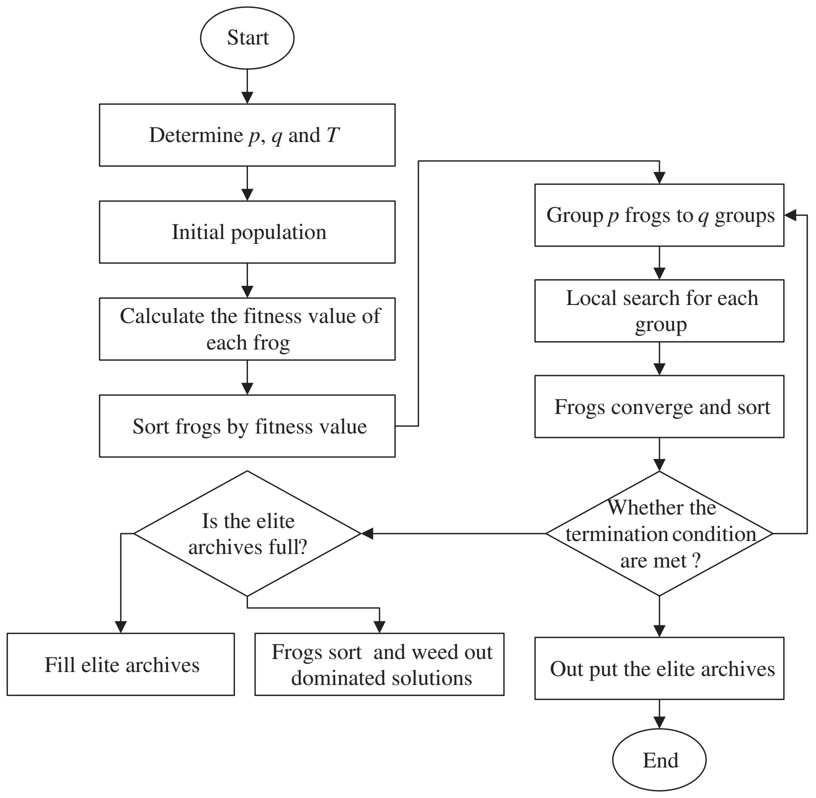
Start
Determine p, q and T
Initial population
Calculate the fitness value of each frog
Sort frogs by fitness value
Group p frogs to q groups
Local search for each group
Frogs converge and sort
Whether the termination condition are met ?
Is the elite archives full?
Fill elite archives
Frogs sort and weed out dominated solutions
Out put the elite archives
End

Figure 3: Flowchart of ILA-CTD

Step 1: Determine the population size p, number of group q, and termination condition T.

Step 2: Initialize population with the proposed method.

Step 3: Calculate the fitness value of each frog in the population.

Step 4: Sort frogs by fitness value.

Step 5: Group p frogs into q groups.

Step 6: Perform local search for each group (update the worst one in each group).

Step 7: Converge frogs in each group, and sort frogs by fitness value again.

Step 8: Whether termination condition is met? If yes, go to Step 9; else, go to Step 10.

Step 9: Are the elite archives full? If the elite archives are not full, then fill elite archives with non-dominated solutions obtained in this generation; else, discard dominated solutions in the elite archives and fill with non-dominated solutions. Then, go to Step 5.

Step 10: Out put the elite archives.

5 A Case Study

The allocation of processing 1,000 pieces of medical labeling machine bottom plates shown in Fig. 4, was used as a case to demonstrate the feasibility of the proposed method. The performance of the plate mainly depends on the dimensional accuracy Q_1, position accuracy Q_2 of the holes, and flatness Q_3 of the plate. The specific accuracy requirements and delivery time constraint is shown in Tab. 1; 10 available services shown in Tab. 2 were obtained after services selection.

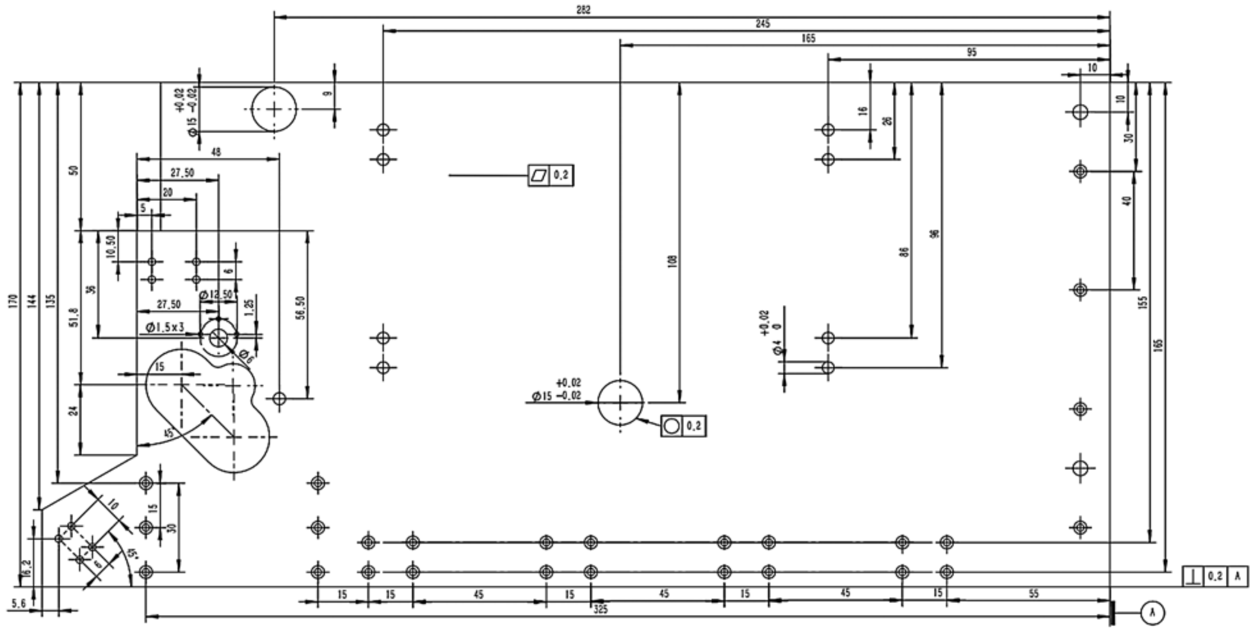

Figure 4: Medical labeling machine bottom plate

Table 1: Requirement information table

Requirement	N	$Tmax/D$	Q_1	Q_2	Q_3
	1000	3	0.8	0.8	0.7

Table 2: Services information table

S^j	$Cma^j/(¥/Pice)$	$Cta^j/(¥/Pice)$	$Tma^j(D/Pice)$	Tta^j	Q^{j1}	Q^{j2}	Q^{j3}	Go^j	Gn^j	Co^j	M^j
1	90	0.8	0.004	0.5	0.90	0.80	0.85	23	22	0.82	100
2	120	0.6	0.002	1.0	0.85	0.90	0.80	125	100	0.78	200
3	115	0.2	0.002	1.5	0.80	0.85	0.70	136	94	0.90	200
4	140	1.2	0.001	0.5	0.95	0.80	0.90	231	128	0.95	500
5	88	1.2	0.005	0.5	0.85	0.80	0.80	45	42	0.88	200
6	100	0.6	0.002	1.0	0.85	0.80	0.80	98	78	0.78	200
7	114	0.4	0.001	1.0	0.85	0.85	0.80	89	76	0.96	300
8	110	1.6	0.002	1.5	0.85	0.80	0.70	79	69	0.76	100
9	115	0.4	0.002	2.0	0.90	0.80	0.75	146	102	0.85	200
10	110	0.5	0.002	1.0	0.80	0.85	0.90	65	59	0.84	200

A personalized solution is always required in actual application, and a comprehensive evaluation based on consumer preferences for targets is necessary. Therefore, the function shown in Eq. (18) was adopted to normalize the attributes values.

$$\begin{aligned}
&\bullet f_a{}^{ij} = \frac{f_a{}^{ij} - f^j{}_{a\min}}{f^j{}_{a\max} - f^j{}_{a\min}} benefit - oriented \\
&\bullet f_a{}^{ij} = \frac{f^j{}_{a\min} - f_a{}^{ij}}{f^j{}_{a\max} - f^j{}_{a\min}} cost - oriented
\end{aligned} \tag{18}$$

$\bullet f_a^{ij}$ is the normalized value of f_a^{ij}; f_a^{ij} is the jth attribute value of the ith available service; f_{amax}^j is the maximum of the jth attribute of available services; f_{amin}^j is the minimum of the jth attribute of available services. In this case, Cma^j, Cta^j, Q^{j1}, Q^{j2}, Q^{j3} and Co^j were normalized first to address their large data fluctuations (Tab. 3).

Table 3: Normalized service attributes

S^j	$Cma^j/(¥/Pice)$	$Cta^j/(¥/Pice)$	$Tma^j(D/Pice)$	Tta^j	Q^{j1}	Q^{j2}	Q^{j3}	Go^j	Gn^j	Co^j	M^j
1	0.9615	0.5714	0.004	0.5	0.6667	0	0.75	23	22	0.3	100
2	0.3846	0.7143	0.002	1.0	0.3333	1	0.5	125	100	0.1	200
3	0.4808	1	0.002	1.5	0	0.5	0	136	94	0.7	200
4	0	0.2857	0.001	0.5	1	0	1	231	128	0.95	500
5	1	0.2857	0.005	0.5	0.3333	0	0.5	45	42	0.6	200
6	0.7692	0.7143	0.002	1.0	0.3333	0	0.5	98	78	0.1	200
7	0.5	0.8571	0.001	1.0	0.3333	0.5	0.5	89	76	0.1	300
8	0.5769	0	0.002	1.5	0.3333	0	0	79	69	0	100
9	0.4808	0.8571	0.002	2.0	0.6667	0	0.25	146	102	0.85	200
10	0.5769	0.7857	0.002	1.0	0	0.5	1	65	59	0.84	200

Table 4: Example solutions of ILA-CTD

S_i	s^1	s^2	s^3	s^4	s^5	s^6	s^7	s^8	s^9	s^{10}	C_i	T_i	Q_i	SI_i	Co_i	G_i
1	0	334	273	0	0	235	0	158	0	0	1.2111	2.046	0.9973	0.997	0.7809	0.9
2	298	0	281	0	0	0	0	0	0	421	1.4466	2.062	1.1942	1.194	0.8614	1.4
3	350	0	0	0	0	360	0	0	0	290	1.4658	2	1.2308	1.23	0.8845	0.8
4	341	486	0	0	0	0	0	0	173	0	1.1566	2	1.4318	1.432	0.8661	0.4
5	0	507	0	0	0	0	0	0	0	493	1.2289	2.014	1.669	1.669	0.8531	0.5
6	199	0	0	507	0	0	0		370	430	1.3874	2.74	1.2676	1.268	0.8401	1.35
7	0	634	0	0	0	0	0	0	0	366	1.1954	2.268	1.7113	1.711	0.8394	0.5
8	190	0	605	0	0	205	0	0	0	0	1.4913	2.71	0.7425	0.743	0.7631	1.1
9	245	489	0	0	0	0	0	0	0	266	1.2754	2	1.6426	1.643	0.867	0.8
10	0	612	0	0	603	0	0	0	0	388	1.2012	2.224	1.704	1.704	0.8418	0.5

Then, the experiment was conducted under the environment of Windows10 and MATLAB 2016a. The parameters of ILA-CTD were set as population size $P = 100$, elite archives = 100, number of groups $q = 5$, evolutionary generation $T = 2{,}000$, speed factors $a = 0.1$ and $b = 10$, and diversity factor $c = 0.4$. The solution set obtained via ILA-CTD is shown in Tab. 4 and Fig. 5.

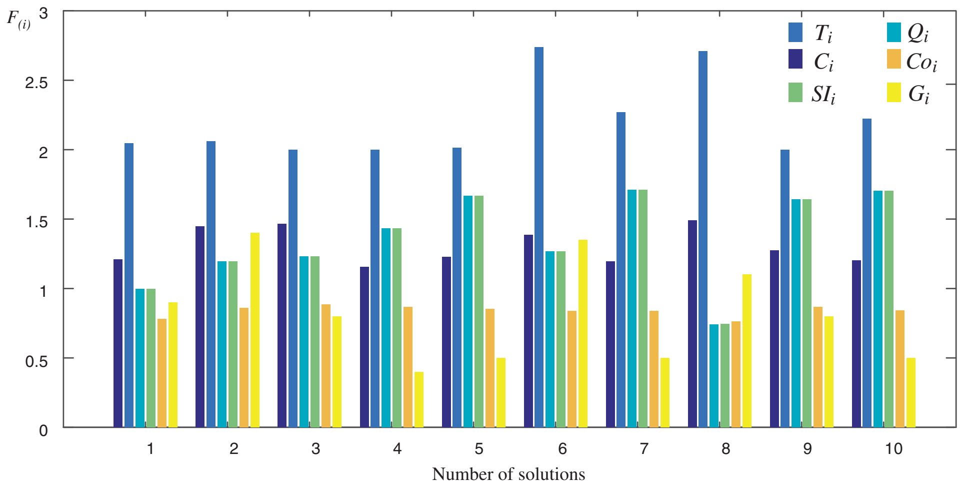

Figure 5: Performances of solutions obtained through ILA-CTD

5.1 *Feasibility Analysis*

The number of non-dominated solutions in the elite archives of each generation (E_d^i shown in Fig. 6) can reflect the dynamic process of the generated non-dominated solutions. As the generations increase, E_a^i grows rapidly at first, and then fluctuates dynamically. This means that better solutions have been found and filled into the elite archives with the increase of generations.

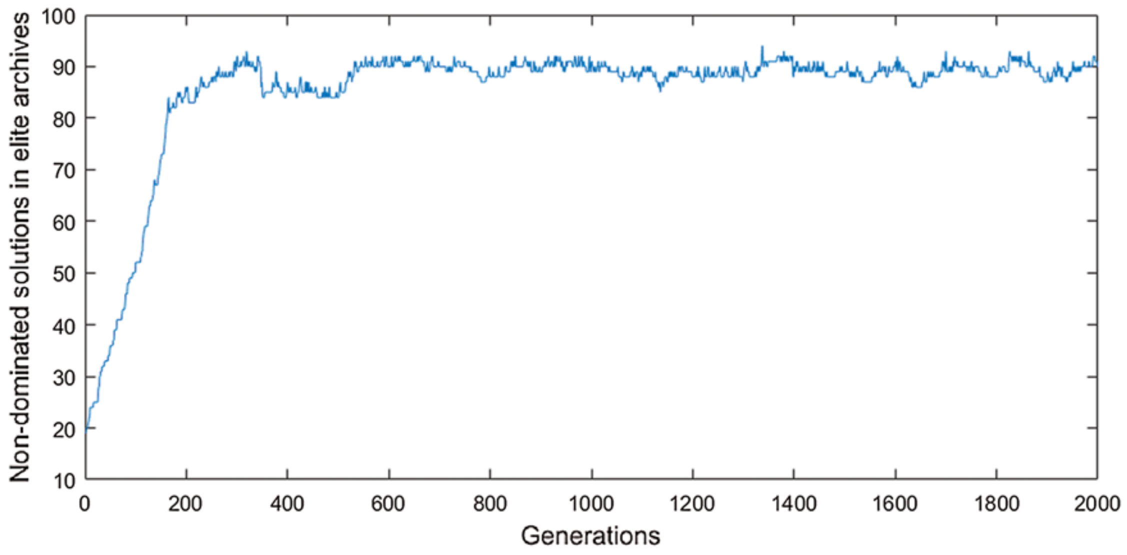

Figure 6: Non-dominated solutions in the elite archives of each generation

The IGD (inverted generational distance) is a simple way to evaluate the convergence of the algorithm based on the optimal solutions set. However, it is impossible for this problem. Because the optimal solutions cannot be obtained at first. Therefore, a new measure was constructed to evaluate the performance of the method proposed in this article. The model proposed in this work is a six-targets problem, and it is difficult to directly display the Pareto frontier. If the algorithm converges, then it means the Pareto frontier converges to a certain area. Thus, the average distance between adjacent generations should become steady. Hence, the average distance between adjacent generations (D_a) was adopted to evaluate the convergence of the proposed method (Eq. (19)).

$$D_a = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n_1} D\left(s_i{}^*, s_j\right)/n_1}{n}$$
$$D\left(s^*{}_i, s_j\right) = \sqrt{\sum_{s=1}^{k}\left(f_i^s - f_j^s\right)^2} \tag{19}$$

$D(s_i^*, s_j)$ is the distance between the ith solution in the kth generation and the jth solution in the $(k+1)$th generation; n_1 and n is the number of non-dominated solutions in the elite archives of the kth and the $(k+1)$th generation, respectively. The evolution process of D_a along generations is shown in Fig. 7.

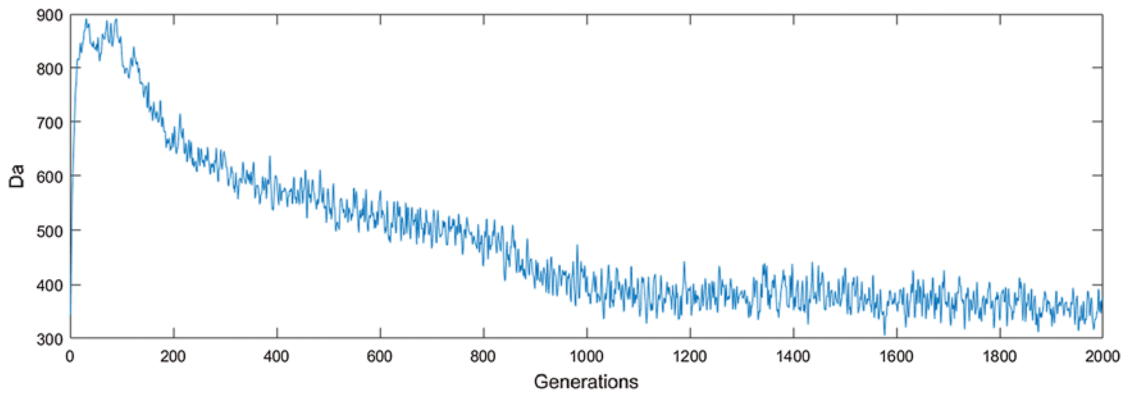

Figure 7: Average distance between adjacent generations

As shown in Fig. 7, D_a rises sharply at first, then falls, and finally stabilizes, which is consistent with the process predicted by the algorithm. The phenomenon results because of three reasons: (1) The non-dominated solutions with different search directions increase dramatically as the generations increase, (2) the optimal search direction is determined progressively and the non-dominated solutions tend to the definite search direction; so, D_a decrease, and (3) the Pareto frontier is found, D_a becomes steady, which validates that the proposed method converges.

5.2 Performance Comparison

To assess the performance of the method proposed in this work, the results obtained via ILA-CTD were compared with MOEA/D-PSO, MOEA/D-GA [11], and NSGA-Π under the same conditions. The solutions set obtained via ILA-CTD, MOEA/D-PSO, MOEA/D-GA and

NSGA-Π are shown in Tabs. 5–7, respectively, and their performances are shown in Figs. 8–10, respectively.

Table 5: Example solutions of MOEA/D-POS

S^j	s^1	s^2	s^3	s^4	s^5	s^6	s^7	s^8	s^9	s^{10}	C_i	T_i	Q_i	SI_i	Co_i	G_i
1	464	0	300	0	0	0	236	0	0	0	1.4758	2.356	1.122	1.122	0.8527	2
2	453	0	0	0	340	0	0	207	0	0	1.251	2.312	0.9941	0.994	0.9314	0.9
3	404	232	0	0	363	0	0	0	0	0	1.341	2.315	1.3	1.298	0.9108	1
4	421	0	0	0	0	305	0	0	274	0	1.4644	2.548	1.1018	1.102	0.8369	0.85
5	0	0	0	0	567	224	209	0	0	0	1.345	3.335	0.9378	0.9378	0.886	1.7
6	0	0	247	0	502	251	0	0	0	0	1.3835	3.01	0.751	0.751	0.839	1.4
7	369	0	221	0	410	0	0	0	0	0	1.4201	2.55	0.9749	0.975	0.8884	1.6
8	500	0	0	0	500	0	0	0	0	0	1.4093	3	1.125	1.125	0.9449	0.9
9	447	0	0	0	318	0	235	0	0	0	1.413	2.228	1.2116	1.211	0.925	1.9
10	254	0	0	501	245	0	0	0	0	0	0.8475	2	1.566	1.566	0.7492	1.85

Table 6: Example solutions of MOEA/D-GA

S^j	s^1	s^2	s^3	s^4	s^5	s^6	s^7	s^8	s^9	s^{10}	C_i	T_i	Q_i	SI_i	Co_i	G_i
1	274	0	0	0	385	0	0	0	341	0	1.3713	2.682	1.0216	1.0216	0.8597	1.35
2	525	0	0	0	475	0	0	0	0	0	1.4155	2.875	1.1396	1.1396	0.9455	0.9
3	0	0	0	573	427	0	0	0	0	0	0.7127	2.635	1.5018	1.5018	0.716	1.55
4	357	0	0	0	367	0	276	0	0	0	1.3937	2.335	1.1796	1.1796	0.9197	1.9
5	0	0	0	0	600	224	176	0	0	0	1.3426	3.5	0.9213	0.9213	0.8886	1.7
6	0	328	0	0	428	0	0	244	0	0	1.0515	2.64	1.0393	1.0393	0.875	0.7
7	475	287	0	0	238	0	0	0	0	0	1.3495	2.4	1.3974	1.3974	0.8884	1.6
8	0	0	0	0	488	0	0	522	0	0	0.9286	2.94	0.58	0.5865	0.9114	0.6
9	392	0	0	0	318	0	280	0	0	0	1.3898	2.09	1.1937	1.1817	0.9109	1.9
10	453	0	0	547	0	0	0	0	0	0	0.8507	2.312	1.7358	1.736	0.7364	1.25

Table 7: Example solutions of NSGA-Π

S^j	s^1	s^2	s^3	s^4	s^5	s^6	s^7	s^8	s^9	s^{10}	C_i	T_i	Q_i	SI_i	Co_i	G_i
1	215	0	238	0	245	0	302	0	0	0	1.4069	2	1.0304	1.03	0.8567	2.6
2	264	0	0	0	580	0	0	0	156	0	1.3591	3.4	0.9914	0.994	0.9314	0.9
3	378	0	0	446	0	176	0	0	0	0	1.341	2.012	1.3	1.300	0.9028	1.35
4	145	287	0	0	0	363	0	0	205	0	1.3505	2.41	1.222	1.222	0.8	0.95
5	236	0	0	0	412	0	352	0	0	0	1.3692	2.56	1.1470	1.147	0.9109	1.9
6	0	342	0	0	500	158	0	0	0	0	1.2531	3	1.1753	1.175	0.866	0.8
7	0	0	261	0	380	369	0	0	0	0	1.4225	2.4	0.7547	0.762	0.8288	1.4
8	500	0	0	0	345	155	0	0	0	0	1.44	2.5	1.125	1.125	0.9236	1
9	221	250	0	0	529	0	0	0	0	0	1.2937	3.145	1.2123	1.212	0.9051	1
10	254	0	0	0	471	0	0	275	0	0	1.1536	2.855	0.844	0.844	0.9227	0.9

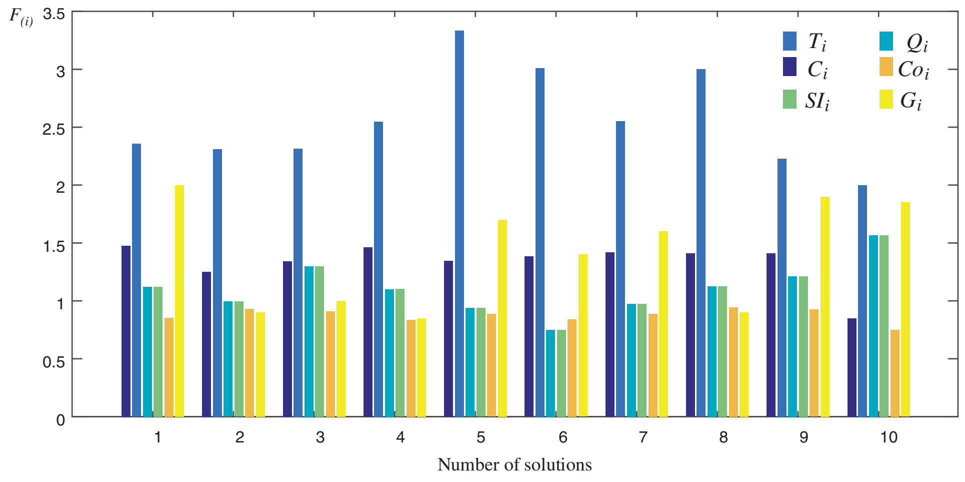

Figure 8: Performances of solutions obtained through MOEA/D-PSO

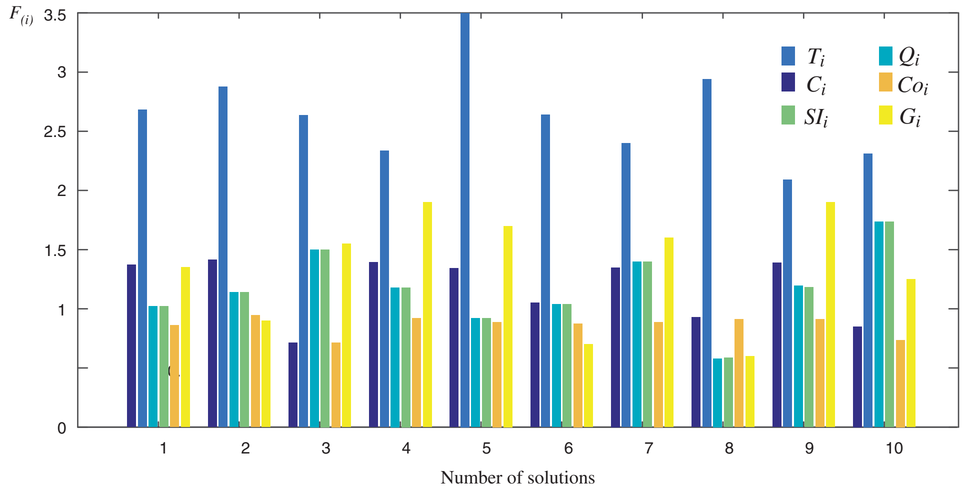

Figure 9: Performances of solutions obtained through MOEA/D-GA

Because MOEA/D-PSO, MOEA/D-GA, and NSGA-Π are methods that have been proven to be feasible in previous studies, they can be used as comparisons to evaluate the performance of ILA-CTD. Two indicators were constructed investigate the relative performance of the proposed method.

1) Proportion of non-dominated solutions in mixed Pareto set t^a

Selecting m solutions from each Pareto set that was finally obtained via MOEA/D-PSO, MOEA/D-GA, NSGA-Π, and ILA-CTD, respectively, and dominance relationships shown in Eq. (20) for the selected $4\,m$ solutions were judged.

$$t^a = \frac{D\left(m^a\right)}{P_D} \tag{20}$$

$$P_D = 4m$$

$D(m^a)$ is the number of non-dominated solutions obtained through method a ($a \in$ MOEA/D-PSO, MOEA/D-GA, NSGA-Π, ILA-CTD), and P_D is total number of solutions in the mixed Pareto set. The comparison results are shown in Fig. 11 ($m = 10$).

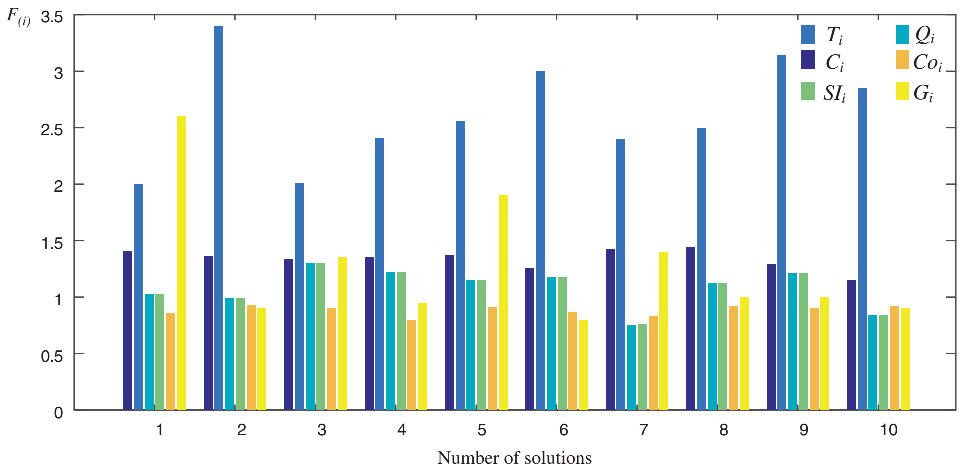

Figure 10: Performances of solutions obtained through NSGA-Π

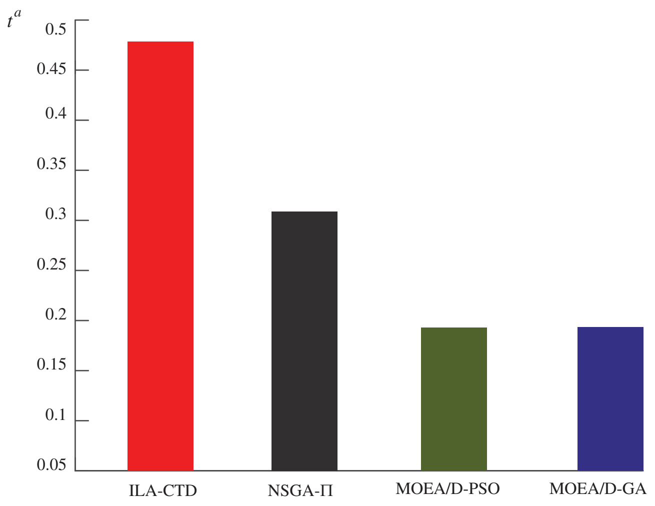

Figure 11: Performance of ILA-CTD in each generation

As the generations increased, more and more non-dominated solutions in P_D were obtained via ILA-CTD. From the perspective of the number of non-dominated solutions in P_D under the same conditions ($T = 2000$ and $m = 10$), ILA-CTD performed best, followed by NSGA-Π, MOEA/D-PSO, and MOEA/D-GA.

2) Proportion of feasible solutions P_f

In practical applications, the obtained solutions must be feasible. Most evolutionary algorithms deal with infeasible solutions through a penalty function. The feasibility of all final solutions cannot be guaranteed if the penalty function is unreasonable; therefore, P_f can express the reliability and practicability of a method. The P_f of ILA-CTD, NSGA-Π, MOEA/D-PSO, and MOEA/D-GA is shown in Fig. 12.

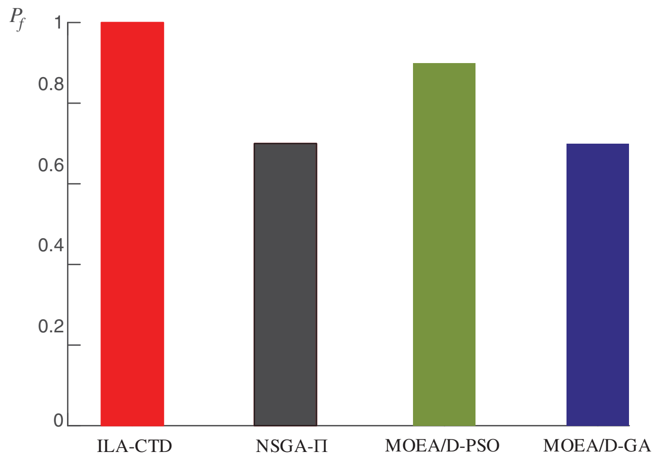

Figure 12: P_f of each method

From the perspective of P_f in the final output solutions, ILA-CTD performed best, followed by MOEA/D-PSO, NSGA-Π, and MOEA/D-GA. It can be concluded that ILA-CTD is more practical compared with the other methods.

6 Conclusions

As one of the research hotspots of CMfg, more and more studies are being undertaken on services selection and their combination especially the combination of heterogeneous services. However, the works related to homogeneous cloud services combinations are relatively few. After reviewing the existing works and noting certain limitations, a homogeneous cloud task distribution model was proposed, and an improved leapfrog algorithm for cloud task distribution (ILA-CTD) was designed to solve the model. The contributions of the paper are summarized as follows:

1) Compared to heterogeneous task distribution, there is relatively little research on homogeneous cloud tasks. A strategy of dealing with homogeneous tasks in the cloud environment was presented to bridge this gap.

2) A specific model was proposed to describe homogeneous cloud task distribution more precisely compared with alternative methods. An improved leapfrog algorithm was designed, which was proven to be more suitable for solving the proposed model.

3) The distribution of real manufacturing tasks of medical labeling machine bottom plates was presented as a case. It was shown that the proposed method provided a combination of factories with a high degree of quality similarity and high informationization under the constraints of factory capacity, task duration, and cost.

To promote the implementation of CMfg platforms with higher agility, future works should focus on the development of efficient demand-service matching algorithm. Personalized service customization should also be studied, because personalized demands from consumers are increasing.

Funding Statement: The research was financially supported by the National Science and Technology Major Project of China (No. 2019ZX04007001), the Science and Technology Major Project of Sichuan Province (No. 2020ZDZX0022).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Zhang, Y., Liu, J., Peng, Y., Dong, Y., Zhao, C. (2020). Performance analysis of intelligent CR-NOMA model for industrial IoT communications. *Computer Modeling in Engineering & Sciences, 125(1),* 239–257. DOI 10.32604/cmes.2020.010778.
2. Xu, S., Chen, J., Wu, M., Zhao, C. (2021). E-commerce supply chain process optimization based on whole-process sharing of internet of things identification technology. *Computer Modeling in Engineering & Sciences, 126(2),* 843–854. DOI 10.32604/cmes.2021.014265.
3. Giret, A., Garcia, E., Botti, V. (2016). An engineering framework for service-oriented intelligent manufacturing systems. *Computers in Industry, 81,* 116–127. DOI 10.1016/j.compind.2016.02.002.
4. Nirmal Kumar, S. J., Ravimaran, S., Alam, M. M. (2020). An effective non-commutative encryption approach with optimized genetic algorithm for ensuring data protection in cloud computing. *Computer Modeling in Engineering & Sciences, 125(2),* 671–697. DOI 10.32604/cmes.2020.09361.
5. Gong, Y., Guo, G. (2019). A data-intensive FLAC3D computation model: Application of geospatial big data to predict mining induced subsidence. *Computer Modeling in Engineering & Sciences, 119(2),* 395–408. DOI 10.32604/cmes.2019.03686.
6. Li, B. H., Zhang, L., Wang, S. L., Tao, F., Cao, J. W. et al. (2010). Cloud manufacturing: A new service oriented networked manufacturing model. *Computer Integrated Manufacturing Systems, 16,* 1–7. DOI 10.13196/j.cims.2010.01.3.libh.004.
7. Li, B. H., Zhang, L., Ren, L., Chai, X. D., Tao, F. et al. (2011). Further discussion on cloud manufacturing. *Computer Integrated Manufacturing Systems, 17,* 451–457. DOI 10.13196/j.cims.2011.03.3.libh.004.
8. Argoneto, P., Renna, P. (2016). Supporting capacity sharing in the cloud manufacturing environment based on game theory and fuzzy logic. *Enterprise Information Systems, 10(2),* 193–210. DOI 10.1080/17517575.2014.928950.
9. Correa, J. E., Toro, R., Ferreira, P. M. (2018). A new paradigm for organizing networks of computer numerical control manufacturing resources in cloud manufacturing. *Procedia Manufacturing, 26(2),* 1318–1329. DOI 10.1016/j.promfg.2018.07.132.
10. Ren, L., Zhang, L., Wang, L., Tao, F., Chai, X. (2017). Cloud manufacturing: Key characteristics and applications. *International Journal of Computer Integrated Manufacturing, 6(6),* 501–515. DOI 10.1080/0951192X.2014.902105.
11. Wu, Y., Jia, G., Cheng, Y. (2019). Cloud manufacturing service composition and optimal selection with sustainability considerations: A multi-objective integer bi-level multi-follower programming approach. *International Journal of Production Research, 58(19),* 6024–6042. DOI 10.1080/00207543.2019.1665203.
12. Maheswari, S., Karpagam, G. R. (2018). Performance evaluation of semantic based service selection methods. *Computers & Electrical Engineering, 71(5),* 966–977. DOI 10.1016/j.compeleceng.2017.10.006.
13. Eisa, M., Younas, M., Basu, K., Awan, I. (2020). Modelling and simulation of QoS-aware service selection in cloud computing. *Simulation Modelling Practice and Theory, 103,* 102108. DOI 10.1016/j.simpat.2020.102108.
14. Chen, Z., Ming, X. (2020). A rough-fuzzy approach integrating best-worst method and data envelopment analysis to multi-criteria selection of smart product service module. *Applied Soft Computing, 94,* 106479. DOI 10.1016/j.asoc.2020.106479.
15. Li, F., Liao, T. W., Zhang, L. (2019). Two-level multi-task scheduling in a cloud manufacturing environment. *Robotics and Computer-Integrated Manufacturing, 56,* 127–139. DOI 10.1016/j.rcim.2018.09.002.

16. Chen, Y., Niu, Y. F., Li, J., Zuo, L. D., Wang, L. (2019). Task distribution optimization for multi-supplier collaborative production in cloud manufacturing. *Computer Integrated Manufacturing Systems, 25,* 1806–1816. DOI 10.13196/j.cims.2019.07.021.
17. Liu, J., Chen, Y. (2019). A personalized clustering-based and reliable trust-aware QoS prediction approach for cloud service recommendation in cloud manufacturing. *Knowledge-Based Systems, 174(1–4),* 43–56. DOI 10.1016/j.knosys.2019.02.032.
18. Eisa, M., Younas, M., Basu, K., Awan, I. (2020). Modelling and simulation of QoS—Aware service selection in cloud computing. *Simulation Modelling Practice and Theory, 103,* 102–108. DOI 10.1016/j.simpat.2020.102108.
19. Zhao, L., Tan, W., Xie, N., Huang, L. (2020). An optimal service selection approach for service-oriented business collaboration using crowd-based cooperative computing. *Applied Soft Computing, 92,* 106270. DOI 10.1016/j.asoc.2020.106270.
20. Hussain, A., Chun, J., Khan, M. (2020). A novel customer-centric methodology for optimal service selection (MOSS) in a cloud environment. *Future Generation Computer Systems, 105,* 562–580. DOI 10.1016/j.future.2019.12.024.
21. Bouzary, H., Chen, F. F. (2020). A classification-based approach for integrated service matching and composition in cloud manufacturing. *Robotics and Computer-Integrated Manufacturing, 66,* 106989. DOI 10.1016/j.rcim.2020.101989.
22. Poordavoodi, A., Goudarzi, M. R. M., Javadi, H. H. S., Rahmani, A. M., Izadikhah, M. (2020). Toward a more accurate web service selection using modifiedinterval DEA models with undesirable outputs. *Computer Modeling in Engineering & Sciences, 123(2),* 525–570. DOI 10.32604/cmes.2020.08854.
23. Devi, R., Shanmugalakshmi, R. (2020). Cloud providers ranking and selection using quantitative and qualitative approach. *Computer Communications, 154,* 370–379. DOI 10.1016/j.comcom.2020.02.028.
24. Gad-Elrab, A. A., Noaman, A. Y. (2020). A two-tier bipartite graph task allocation approach based on fuzzy clustering in cloud-fog environment. *Future Generation Computer Systems, 103(11),* 79–90. DOI 10.1016/j.future.2019.10.003.
25. Gigliotta, O. (2018). Equal but different: Task allocation in homogeneous communicating robots. *Neurocomputing, 272(3),* 3–9. DOI 10.1016/j.neucom.2017.05.093.
26. Sharma, V., Bala, M. (2020). An improved task allocation strategy in cloud using modified k-means clustering technique. *Egyptian Informatics Journal, 21(4),* 201–208. DOI 10.1016/j.eij.2020.02.001.
27. Moussa, M., ElMaraghy, H. (2020). A genetic algorithm-based model for product platform design for hybrid manufacturing. *Procedia CIRP, 93,* 389–394. DOI 10.1016/j.procir.2020.04.044.
28. Jiang, H., Yi, J., Chen, S., Zhu, X. (2016). A multi-objective algorithm for task scheduling and resource allocation in cloud-based disassembly. *Journal of Manufacturing Systems, 41(4),* 239–255. DOI 10.1016/j.jmsy.2016.09.008.
29. Jatoth, C., Gangadharan, G. R., Fiore, U. (2019). Optimal fitness aware cloud service composition using modified invasive weed optimization. *Swarm and Evolutionary Computations, 44(4),* 1073–1091. DOI 10.1016/j.swevo.2018.11.001.
30. Zhou, J., Gao, L., Yao, X., Zhang, C., Chan, F. T. et al. (2019). Evolutionary algorithms for many-objective cloud service composition: Performance assessments and comparisons. *Swarm and Evolutionary Computation, 51,* 100605. DOI 10.1016/j.swevo.2019.100605.
31. Somasundaram, T. S., Govindarajan, K. (2014). CLOUDRB: A framework for scheduling and managing high-performance computing (HPC) applications in science cloud. *Future Generation Computer Systems, 34(3),* 47–65. DOI 10.1016/j.future.2013.12.024.
32. Guo, Y., Tian, X., Fang, G., Xu, Y. (2020). Many-objective optimization with improved shuffled frog leaping algorithm for inter-basin water transfers. *Advances in Water Resources, 138,* 103531. DOI 10.1016/j.advwatres.2020.103531.

33. Shampine, L. F. (2009). Stability of the leapfrog/midpoint method. *Applied Mathematics and Computation, 208,* 293–298. DOI 10.1016/j.amc.2008.11.029.

34. Manimegalai-Sridhar, U., Govindarajan, A., Rhinehart, R. R. (2014). Improved initialization of players in leapfrogging optimization. *Computers & Chemical Engineering, 60(2),* 426–429. DOI 10.1016/j.compchemeng.2013.08.009.