A Reinforcement Learning-Variable Neighborhood Search Method for the Cloud Manufacturing Scheduling Robust Optimization Problem with Uncertain Service Time

Sihan Wang(✉) and Chengjun Ji

School of Business Administration, Liaoning Technical University, Huludao 125100, Liaoning, China
2456020576@qq.com

Abstract. Cloud manufacturing (CMfg) is an advanced networked intelligent manufacturing model, which includes a large number of new product customization services. Since many products lack historical data on service time, there is uncertainty about CMfg product service time, thus, CMfg service platforms need to perform robust scheduling of CMfg services for new products. In this paper, a CMfg scheduling model considering service time uncertainty and non-predefined service paths is constructed, and its robust equivalent is derived. In order to effectively solve the above model, this paper proposes a reinforcement learning-variable neighborhood search algorithm (rVNS) based on the variable neighborhood search algorithm, in which the upper confidence bound algorithm (UCB1) is used to adaptively select the neighborhood operator. To solve the problem of insufficient historical data at its cold start, the SARSA (lambda) method is used in this paper. In addition, this paper leverages adaptive windows to estimate and detect changes in rewards in data streams to obtain more accurate reward estimates. A large number of experiments prove that the algorithm designed in this paper has high accuracy and speed advantages in solving this problem.

Keywords: CMfg scheduling · robust optimization · non-predefined service paths · variable neighborhood search algorithm · upper confidence bounds algorithm · reinforcement learning

1 Introduction

CMfg is an advanced networked intelligent manufacturing model, which includes a large number of new product customization services [1]. In the past few years, the research on the optimization of CMfg services has mainly focused on service combination and the scheduling of CMfg tasks, Lim et al. [2] studied the service combination and optimization selection problem considering the interests of service demanders, cloud platform operators and resource providers. Laili [3] studies the multi-stage integrated scheduling of hybrid tasks in the CMfg environment. Variable neighborhood search (VNS) is a well-known meta-heuristic originally proposed by mladenovic and Hansen in 1997 [4],

© The Author(s) 2024
S. H. B. D. M. Zailani et al. (Eds.): ICMSEM 2023, 259, pp. 524–533, 2024.
https://doi.org/10.2991/978-94-6463-256-9_54

VNS has evolved to solve various optimization problems. VNS is well scalable and can be extended to various optimization problems. In the field of intelligent optimization, scholars have demonstrated that more promising techniques can be obtained using a mixture of reinforcement learning and intelligent algorithms [5, 6].

Through the above analysis, the traditional scheduling and service combination research has not fully studied customized products. Since many customized products such as mobile phone models [7] lack historical service data, and service time estimation can only be based on similar products, we use robust optimization to solve such problems. In addition, different decomposition methods will produce different service paths, increasing the flexibility of CMfg systems by considering non-predefined path approaches. Therefore, this paper decomposes the task into subtasks through different service paths. In terms of the solution method, the effect of combining the upper confidence bounds (UCB1) algorithm [8] and meta-heuristics are studied for the first time. The reinforcement learning-variable neighborhood search algorithm (rVNS) is designed, which utilizes the upper confidence bounds algorithm (UCB1) for adaptive neighborhood selection. To solve the problem of insufficient historical data, the SARSA (λ) method is used [9], in addition, the adaptive window is utilized for detecting sudden changes in reward distribution and restarting UCB tasks [10], which is a state-of-the-art algorithm to estimate and detect changes in reward in data streams.

2 Optimization Model

2.1 Problem Description

We build a robust optimization model for CMfg scheduling with uncertain service time in new customized products. In the CMfg environment, N service providers with K manufacturing services located in different geographical regions are considered. There are alternative service paths in the CMfg system. Each service provider may offer multiple service types. The service demander submits T tasks to the cloud platform, and each CMfg order can be split into several CMfg subtasks, each subtask is served by different CMfg service providers with same service type, we assume that there is a linear processing sequence between the suborders of the same order. Each task must be completed by a given due date. Different service providers provide different unit costs and required service times for specific subtasks. This article is from the perspective of a cloud manager who has the authority to choose an appropriate service path to minimize the total duration. In addition, we considers balancing the load of service providers so that CMfg services can be carried out smoothly.

2.2 Mathematical Model

The robust optimization model of CMfg scheduling under uncertain machining time constructed in this paper uses the following symbols as shown in Table 1.

The objective function of the model is to minimize the makespan and reduce the load on the CMfg service, as shown in Eq. (1):

$$\min f = 1 - \alpha\left(\max C_{ijl}\right) + \alpha \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{t=1}^{l} \sum_{k=1}^{h} \tilde{P}_{\mathrm{ijtk}} \mathrm{X}_{\mathrm{ijtk}} \tag{1}$$

Table 1. Variables and parameters of the CMfg task scheduling model

Symbol	Description	Symbol	Description
n	The number of CMfg tasks	T_{ji}	The set of CMfg subtasks of the path j of CMfg task i
m	The number of CMfg task paths	K_{tji}	The set of service providers for the t CMfg subtask of the path j of CMfg task i
l	The number of CMfg subtasks	$\tilde{p}_{ktji}$	The service time for the CMfg subtask t of the path j of CMfg task i is provided by CMfg service provider k.
h	The number of CMfg service providers	$O_{kk'}$	The distance between Service Provider k and Service Provider k'
I	The set of CMfg tasks $i \in I, I = \{1, 2, \cdots, n\}$	d_i	The latest delivery date for the task i.
J	The set of CMfg task paths $j \in J, J = \{1, 2, \cdots, m\}$	S_{tji}	The start time of the processing of the subtask t of the path j of the task i.
T	The set of CMfg subtasks $t \in T, T = \{1, 2, \cdots, l\}$	C_{tji}	The end time of the processing of the subtask t of the path j of the task i.
K	The set of CMfg service providers $k \in K, K = \{1, 2, \cdots, h\}$	x_{ktji}	1 if the subtask t of the path j of task i is processed by service provider k, otherwise 0
J_i	The set of CMfg task paths of CMfg task i.	y_{ji}	1 if the task i is processed by the path j, otherwise 0

s.t.

$$\sum_{k \in h} x_{ktji} = 1 \quad \forall i \in I, j \in J, t \in T \tag{2}$$

$$S_{tji} - S_{(t-1)ji} \geq \sum_{k \in K} \tilde{P}_{k'(t-1)ji} x_{k'(t-1)ji} + \sum_{k,k' \in K} O_{\mathrm{kk'}} \mathrm{X}_{\mathrm{ij(t-1)k'}} X_{ijtk} \forall j \in J, \forall t \in T \tag{3}$$

$$\left| S_{tji} - S_{t'j'i'} \right| \geq x_{ktji} x_{kt'j'i'} \theta, \text{ where } \theta = \begin{cases} p_{kt'j'i'}, \text{ if } S_{tji} - S_{t'j'i'} \geq 0 \\ p_{ktji}, \text{otherwise} \end{cases}, \forall i \in I, \forall k \in K \tag{4}$$

$$\sum_{j \leq J_i} y_{ji} = 1 \quad \forall i \tag{5}$$

$$S_{ktji} \leq d_i \quad \forall i, j \in J_i, t \in T_{ji}, k \in K_{tji} \tag{6}$$

$$C_{tij} = s_{tij} + \sum_{k \in K} \tilde{P}_{ktij} x_{ktij}, \quad \forall i \in I, \forall j \in J, \quad \forall t \in T \tag{7}$$

$$s_{(t+1)ij} - C_{tij} \geq O_{kk'} x_{k'(t+1)ij} x_{ktij} \quad \forall i \in I, \forall j \in J, \quad \forall t \in T, \forall k \in K \tag{8}$$

Constraints (2) Ensure that each CMfg task can only be assigned to one available CMfg service provider. Constraint (3) is a subtask prioritization constraint, that is, a subtask belonging to the same CMfg task must start after the subtask that precedes it completes. Constraints (4) ensure that the service provider handles only one subtask at a time. Constraint (5) means that only one route can be selected for each task. Constraint (6) ensure that manufacturing tasks must be completed before due dates. Constraint (7) specifies the completion time of the CMfg subtask. Constraint (8) means that if the task is assigned to different CMfg service providers to service, transportation is required.

2.3 Scheduling Model Based on Robust Optimization.

Due to the uncertain processing time of service providers, this paper considers the use of budget set and box uncertain set to characterize service time.

For uncertain parameters ξ_{ij}, The uncertain set is expressed as Formula (9).

$$U = \left\{ \xi_{ij} \left| \sum_{j} \right| \left| \xi_{ij} \right| \leq \Gamma_i, \left| \xi_{ij} \right| \leq 1, \forall i, j \in J_i \right\} \tag{9}$$

The robust dual transformation yields the Eqs. (10)–(13), where μ_{ij} and z_i are the auxiliary decision-making variable.

$$f - \sum_{i}^{ni} \sum_{j=1}^{nj} \sum_{t=1}^{ns} \left[\sum_{k=1}^{nk} \left(\hat{P}_{ijtk} + \Gamma_k z_k \right) x_{ijtk} + \mu_{ijt} \right] \geq 0 \tag{10}$$

$$S_{tji} - S_{(t-1)ji} \geq \sum_{k \in K} \left(\hat{P}_{k(t-1)ji} + \Gamma_k z_k + \mu_{ijt} \right) x_{k(t-1)ji} \tag{11}$$

$$\left| S_{tji} - S_{t'j'i'} \right| \geq x_{ktji} x_{kt'j'i'} \theta, \text{where} \, \theta = \begin{cases} \hat{P}_{\mathrm{kt'j'i'}} + \Gamma_{\mathrm{k}} z_{\mathrm{k}} + \mu_{t'j'i'}, \text{if } \mathrm{S_{tji}} - \mathrm{S_{t'j'i'}}^{3} 0 \\ \hat{P}_{\mathrm{ktji}} + \Gamma_{\mathrm{k}} z_{\mathrm{k}} + \mu_{\mathrm{tij}}, \text{otherwise} \end{cases} \tag{12}$$

$$C_{tij} = s_{tij} + \sum_{k \in K} \left(\hat{P}_{ktij} + \Gamma_k z_k + \mu_{ijt} \right) x_{ktij}, \quad \forall i \in I, \forall j \in J, \quad \forall t \in T \tag{13}$$

(2), (5), (6), (8).

3 A Reinforcement Learning-Variable Neighborhood Search Algorithm

3.1 General Variable Neighborhood Search Algorithm

General variable neighborhood search is a local search algorithm, which is achieved by systematically changing the neighborhood structure in the search process, the general variable neighborhood search algorithm includes shaking and variable neighborhood descent (VND). Shaking is used to jump out of the local optimum, VND is used to find the local optimal through the neighborhood transformation of the system.

3.2 Encoding and Decoding

In this paper, a three-layer coding method is used to encode the CMfg service provider and processing path corresponding to each CMfg subtask of each CMfg task [2].

3.3 Reinforcement Learning Strategy

It is very important to balance exploration and development in heuristic algorithms, which have a significant impact on both accuracy and convergence speed, and the GVNS chooses the appropriate neighborhood structure at a certain point in the search or after improvement depends on many factors and determines the performance of the algorithm framework. Decisions are made based on the knowledge that intelligent algorithms have gained from previous actions. The agent needs to discover strategies to minimize expected regret or maximize its reward. We select the UCB1 algorithm in the UCB family, which is derived from the Hoeffding inequality. This paper applies UCB1 to the GVNS metaheuristic, the agent's strategy is to select a neighborhood action in each iteration t that maximizes the following criteria:

$$A_t = \underset{j=\{1\ldots l\}}{\arg\max}[\hat{p_t}(a_j) + c\sqrt{\frac{2\log\sum_{l=1}^{L} n_l}{n_t(a_j)}}] \tag{14}$$

The right part of Eq. (14) consists of two distinct parts. The first part represents the average experience reward obtained, a_j is the jth neighborhood action with a total of L neighborhoods, t is the number of iterations, and the number of iterations of the lneighborhood action l is n_l. Thus, the first part represents the expected reward for the response to the chosen neighborhood action. The second part represents exploration, $n_t(a_j)$ indicating the number of times the jth neighborhood action are used before the t moment, and C is the scaling factor required to balance the trade-off between exploitation and exploration. Traditional Q learning and SARSA only update the Q value of the previous step at a time. However, the choice of each action has a real impact on the final reward. Therefore, this study combines Q-learning and SARSA and historical tracking (ET) in the rVNS algorithm.

3.4 Neighbourhood Actions

We defines four neighborhood actions for the above coding methods. (1) Relocate. This operator deletes and replaces a node from one code, and then inserts it into the same code. (2) Exchange. The exchange operator swaps the positions of two nodes. (3) Swap (λ_1, λ_2). This operator swaps λ_1 consecutive indexes in one code with λ_2 consecutive indexes in the same code. (4) Shift (λ_1, 0). This operator removes λ_1 consecutive indexes and inserts them into another position in the same code.

3.5 Reward Data Detection

Over time, the underlying relationships in the data change unexpectedly. This phenomenon is known as conceptual drift. The UCB algorithm makes decisions based partly on rewards. In dynamically changing and non-static environments, the expected reward distribution may change, so the UCB algorithm must dynamically adjust its estimates. We uses ADWIN [11] to reset the learning process in a major conceptual drift event. The algorithmic pseudocode as shown in algorithm 1, where W is the time window length. $\hat{\mu}_{W_0}$ and $\hat{\mu}_{W_1}$ is the estimated mean of the segmented fragment, the value of ε as shown in the Eq. (15).

$$\varepsilon = \sqrt{\frac{1}{2\frac{1}{1/n_0} + \frac{1}{1/n_1}} \times ln\frac{4n}{\delta}} \tag{15}$$

where n_0 and n_1 are the length of the two fragments of data after splitting the time window, n is the total time window length, δ is the confidence value.

Algorithm 1: ADWIN

Input: Time window length w, reward data

Output: $\hat{\mu}_W$ reward estimates in the time window

1. For each iteration t do;
2. $W \leftarrow W \cup \{x_t\}$;
3. Repeat Delete data from the W tail;
4. Until $|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}| \geq \varepsilon$ For each split in the time window $W = W_0 + W_1$

3.6 RVNS Algorithm Process

The pseudo code of the rVNS algorithm designed in this paper is shown in algorithm 2.

Algorithm 2: rVNS

Input: x_{init} :current solution, kmax :maximum number of neighborhoods, tmax :maximum run time, w: sliding time window

Output: Optimal solution x

1. Generate an initial solution x_{init};
2. k←1;
3. **While** $t \le t_{\max}$ and $step \le step_{\max}$ **do**
4. x'←Shake(x_{init},kmax)//Given the initial solution, a neighborhood action is performed.
5. **While** *true* **do**
6. im ←false;
7. k=k+1 ;
8. **If** Data drift detected **then**
9. reset (k,p) ;
10. l ←rand(0,$k_{\max}$);
11. **else**
12. l← $UCB1(c, step, k, p)$
13. **End If**
14. **If** $f(x'') \le f(x')$ **then**
15. $x \leftarrow x''$;
16. $im \leftarrow true$;
17. **End If**
18. reward← $f(x') - f(x'')$
19. Update Q table//based on Q-learning(λ)and SARSA(λ)
20. **If** not improve **then**
21. **Break**
22. **End If**
23. **End While**
24. t←Cputime();
25. step←step+1;
26. **End while**
27. **return** x

4 Simulation Experiments

4.1 Experiments Design

In order to verify the effectiveness of the proposed method, two comparative experiments are designed, Table 2 describes the data values and distributions used by the algorithm, and some parameters are taken from uniform distributions, so 10 experimental data are generated. These simulation experiments of different scales are selected to evaluate the processing time to find the best scheduling plan in each case, and the genetic algorithm, ant colony algorithm and simulated annealing algorithm are compared.The simulation environment in this article is a 2 GHz Inter® Core™i7 CPU and 8 GB RAM, programmed using MATLAB.

Table 2. Parameters used by the algorithm

Contents	Value/distribution
Number of paths for each task	U(1,10)
Number of sub-tasks for each task	U(4,20)
Subtasks service time matrix	U(4,20)
Logistics time matrix	U(1,5)
time window length w	10
tasks due date matrix	U(30,600)
c	U(1,2)
α	U(0,1)

4.2 Experimental Results Analysis

In each example, each algorithm runs 10 times in this section to solve the model proposed in this article, the solution time of each algorithm is limited to 1h, and in 10 runs, all algorithms obtain the understanding set in a limited time, indicating that all algorithms have good feasibility. Table 3 compares the optimization results of five algorithms. It can be seen from the experimental results that the rVNS algorithm proposed in this paper is better than GA, ACO, SA and VNS in terms of solution performance, and with the increase of the scale of the study, the solution performance advantages of the proposed rVNS algorithm in this paper become more obvious. The above experimental results show that the rVNS algorithm has a better optimization effect for solving the robust optimization problem of CMfg scheduling.

5 Conclusions

Because CMfg products have a high degree of personalization, the manufacturing resources and service history data required by different parts of these customized products are insufficient, we build a robust optimization model for CMfg scheduling with uncertain service time, and design a rVNS algorithm to slove the problem, in which the upper confidence bounds algorithm (UCB1) is used. In order to solve the problem of insufficient historical data at algorithm start, we use the SARSA (λ) method, in addition, adaptive window, adaptive window algorithm is used to estimate and detect changes in rewards in data streams. We prove the advantages of the algorithm designed in this paper in accuracy and speed through a large number of experiments.

Table 3. Comparison of the 5 optimization algorithms

Case	GA		ACO		SA		VNS		rVNS	
	Time (s)	Obj	Time (s)	Obj	Time (s)	Obj	Time (s)	Obj	Time (s)	Obj
1	2.1968	**41.2**	1.8784	**41.2**	1.1576	**41.2**	0.8328	**41.2**	**0.532**	**41.2**
2	3.4176	**76.08**	2.6904	**76.08**	2.8984	**76.08**	1.4776	**76.08**	**0.9312**	**76.08**
3	4.1336	103.36	3.5168	103.36	3.8544	103.36	2.4568	**101.92**	**2.0784**	**101.92**
4	4.6592	138.64	3.6216	138.64	4.1256	**136.24**	6.9008	137.2	**3.788**	**136.24**
5	12.1232	229.76	11.4072	231.6	9.3232	229.6	11.1728	221.6	**7.8632**	**221.36**
6	15.5968	348.96	16.5536	350	17.2448	375.36	24.1456	334.72	**15.528**	**320.96**
7	28.6216	791.76	52.8104	770.24	22.4504	739.52	31.288	659.84	**21.5248**	**621.12**
8	62.2576	1201.6	60.1592	1223.28	62.9368	1263.36	58.8168	1179.84	**60.8784**	**1103.12**
9	90.7624	1868	104.3296	1952	76.8856	1912.32	68.3352	1625.36	**61.7616**	**1616.48**
10	172.5976	2382.24	200.1848	3112.88	215.7304	2360.88	282.7512	1968.08	**163.968**	**1841.36**

References

1. Li B. H., Zhang L., Wang S. L., et al. (2010). Cloud manufacturing: a new service-oriented networked manufacturing model [J]. Computer integrated manufacturing systems, 16(1): 1–7. DOI: https://doi.org/10.13196/j.cims.2010.01.3.libh.004
2. Lim M. K., Xiong W., Wang Y. (2022). A three-tier programming model for service composition and optimal selection in cloud manufacturing [J]. Computers & Industrial Engineering, 167: 108006. DOI: https://doi.org/10.1016/j.cie.2022.108006.
3. Laili Y., Lin S., Tang D. (2020). Multi-phase integrated scheduling of hybrid tasks in cloud manufacturing environment [J]. Robotics and Computer-Integrated Manufacturing, 61: 101850. DOI: https://doi.org/10.1016/j.rcim.2019.101850.
4. Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. Computers & Operations Research, 24(11), 1097–1100. https://doi.org/10.1016/S0305-0548(97)00031-2.
5. Sun, Z., Benlic, U., Li, M., & Wu, Q. (2022). Reinforcement learning based tabu search for the minimum load coloring problem. Computers & Operations Research, 143, Article 105745. DOI: https://doi.org/10.1016/j.cor.2022.105745.
6. Wauters, T., Verbeeck, K., De Causmaecker, P., & Berghe, G. V. (2013). Boosting metaheuristic search using reinforcement learning. In Hybrid metaheuristics (pp. 433–452). Springer. DOI: https://doi.org/10.1007/978-3-642-30671-6-17.
7. Xu, X., Cui, W., Lin, J., and Qian, Y. (2013). Robust makespan minimisation in identical parallel machine scheduling problem with interval data. International Journal of Production Research, 51(12):3532–3548. DOI: https://doi.org/10.1080/00207543.2012.751510.
8. Auer, P., Cesa-Bianchi, N., & Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. Machine Learning, 47(2–3), 235–256. https://ieeexplore.ieee.org/document/7744106/
9. E. Pakizeh, M.M. Pedram, M. Palhang. (2015). Multi-criteria expertness based cooperative method for SARSA and eligibility trace algorithms. Appl. Intell. 43: 487–498, https://doi.org/10.1007/s10489-015-0665-y.
10. Bifet, A., & Gavalda, R. (2007). Learning from time-changing data with adaptive windowing. In Proceedings of the 2007 SIAM international conference on data mining (pp. 443–448). SIAM. http://www.researchgate.net/profile/Albert_Bifet/publication/220907178_Learning_from_Time-Changing_Data_with_Adaptive_Windowing/links/0deec520f3bb300773000000
11. Montiel, J., Read, J., Bifet, A., & Abdessalem, T. (2018). Scikit-multiflow: A multi-output streaming framework. Journal of Machine Learning Research, 19(72), 1–5. DOI: https://doi.org/10.5555/3291125.3309634.

Open Access This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial 4.0 International License (http://creativecommons.org/licenses/by-nc/4.0/), which permits any noncommercial use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.