TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 22/31 pp303–317
DOI: 10.26599/TST.2024.9010012
Volume 30, Number 1, February 2025

Time-of-Use Price Resource Scheduling in Multiplex Networked Industrial Chains

Pan Li†, Kai Di†, Xinlei Bai, Fulin Chen, Yuanshuang Jiang, Xiping Fu, and Yichuan Jiang*

Abstract: With the advancement of electronic information technology and the growth of the intelligent industry, the industrial sector has undergone a shift from simplex, linear, and vertical chains to complex, multi-level, and multi-dimensional networked industrial chains. In order to enhance energy efficiency in multiplex networked industrial chains under time-of-use price, a coarse time granularity task scheduling approach has been adopted. This approach adjusts the distribution of electricity supply based on task deadlines, dividing it into longer periods to facilitate batch access to task information. However, traditional simplex-network task assignment optimization methods are unable to achieve a globally optimal solution for cross-layer links in multiplex networked industrial chains. Existing solutions struggle to balance execution costs and completion efficiency in time-of-use price scenarios. Therefore, this paper presents a mixed-integer linear programming model for solving the problem scenario and two algorithms: an exact algorithm based on the branch-and-bound method and a multi-objective heuristic algorithm based on cross-layer policy propagation. These algorithms are designed to adapt to small-scale and large-scale problem scenarios under coarse time granularity. Through extensive simulation experiments and theoretical analysis, the proposed methods effectively optimize the energy and time costs associated with the task execution.

Key words: multiplex networked industrial chains; resource scheduling; time-of-use price

1 Introduction

Multiplex networked industrial chains systems have incorporated various heterogeneous intelligent components to manage diverse production tasks. These intelligent agents exhibit dissimilar collaborative relationships, thereby introducing new characteristics to the multi-network environment. However, applying traditional simplex network scheduling algorithms directly to multiplex networked industrial production is limited. They compute locally optimal scheduling schemes for each network layer based on dynamic factors like production cost only. Consequently, this approach fails to consider task constraints across multiplex networked, resulting in suboptimal scheduling outcomes for the entire system. Simultaneously, most industrial production functions consume substantial electricity in multiplex networked industrial systems. Electricity suppliers have recently implemented a time-of-use (TOU) pricing system, leading to notable fluctuations in electricity prices

• Pan Li, Xinlei Bai, and Fulin Chen are with School of Cyber Science and Engineering, Southeast University, Nanjing 211189, China. E-mail: lipan@seu.edu.cn; xlbai@seu.edu.cn; chenfulin@seu.edu.cn.
• Kai Di, Yuanshuang Jiang, and Yichuan Jiang are with School of Computer Science and Engineering, Southeast University, Nanjing 211189, China. E-mail: dikai@seu.edu.cn; yuanshuangjiang@seu.edu.cn; yjiang@seu.edu.cn.
• Xiping Fu is with the PredictHQ Ltd., Auckland 1010, New Zealand. E-mail: xiping@predicthq.com.
• A preliminary version of this paper was presented at the 24th Internationa conference on Parallel and Distributed Computing, Applications and Technologies[1]
† Pan Li and Kai Di contribute equally to this work.
* To whom correspondence should be addressed.
Manuscript received: 2023-11-13; revised: 2023-12-27; accepted: 2024-01-04

during different periods. Consequently, optimizing task deployment tailored to multi-chain industrial systems becomes crucial in rationalizing production task execution at varying time intervals to reduce operational costs associated with production agents.

In the context of coarse time granularity task allocation, the allocation calculation for specific tasks is divided into more extended periods. Multiplex tasks can be included and allocated simultaneously or sequentially within these periods, following a particular strategy. Coarse time granularity task allocation is generally suitable for scenarios with relatively low task response time requirements. In these scenarios, the optimal allocation plan for a given stage is determined by considering all relevant information collected over an extended period[2]. Offline task allocation represents a typical scenario for coarse time granularity task allocation, where the global optimal task allocation can be achieved by batch acquisition of task information. Therefore, this research focuses on addressing the problem of optimizing coarse time granularity task allocation in a multi-chain industrial task processing system. Building upon the multiplex networked Industrial chains characteristics in the task processing system, a task allocation optimization mechanism is proposed under coarse time granularity. This mechanism ensures the reasonable arrangement of execution product agents and specifies the execution time of tasks across different industrial chains based on the electricity price during different periods. The objective is to minimize the maximum completion time and required electricity cost for task execution.

Coarse time granularity task allocation optimization in a multiplex networked industrial chains task processing system encounters two main challenges. The first challenge is handling the impact of multiplex networked industrial chains on task allocation. Specifically, it involves modeling the correlation between tasks in each industrial chain within multiplex networked industrial chains. The second challenge is to construct a suitable task allocation scheme. This scheme aims to optimize the execution cost and execution time of tasks while satisfying various constraint conditions.

In the past, task allocation optimization typically only considered the selection of product agents that execute tasks within a simplex industry chain and the allocation of specific execution times for each task. Optimization allocation problems were modeled as single-machine scheduling, workshop scheduling, flow-shop scheduling, and other issues, and were then optimized and solved through corresponding algorithms[3, 4]. However, with the gradual convergence of industrial chains and the trend towards multiplex industrial chains, previous methods have exposed some limitations. In other words, traditional methods did not consider the inherent relationships between different tasks in different industrial chains, resulting in poor performance when directly applied to multiplex networked industrial chains.

This paper investigates the relationship between tasks and multiplex networked industry chains. Specifically, it focuses on studying the impact of multiplex networked industry chains on task deployment and optimizing task allocation for each industry chain at the system level. The aim is to achieve a rational allocation of tasks across industry chains, to optimize the energy cost and execution time of task processing at the overall level. This will improve the execution efficiency of the task processing system.

The main contributions of this paper are summarized as follows:

(1) The modeling and analysis of task execution in the task processing system under multiplex networked industry chains is proposed, and a mathematical model for a task processing system adapted to multiplex networked industry chains is presented. This model focuses on expressing the relationship between tasks in different industry chains under multiplex networked industry chains and provides a foundation for subsequent task deployment arrangements.

(2) For the case of small-scale data with fewer network layers, a mixed integer linear programming algorithm based on branch-and-bound is proposed. A fast-pruning strategy is designed for the characteristics of multiplex network industry chains to prune the search space during initialization and reduce the complexity of subsequent searches. For the case of large data volumes and complex industry chain structures, a heuristic cascading adaptive deployment algorithm based on cross-chain strategy propagation is proposed, involving multiple rounds of iterative deployment.

(3) The performance of the algorithm proposed in this article has been verified through many simulation

experiments and compared with traditional deployment optimization algorithms under a simplex industry chain. Through theoretical analysis and experimental validation, it is shown that the algorithm proposed in this article has better results under multiplex networked industry chains, and the energy cost and execution time of tasks have been optimized to a certain extent.

The structure of the paper is organized as follows. Section 2 provides a literature review. Section 3 describes the coarse time granularity allocation optimization problem and its mathematical model of multiplex networked industrial chain task processing systems in detail. Section 4 introduces the proposed algorithm for small-scale problem scenarios. Section 5 presents the proposed solution algorithm for large-scale problem scenarios. Section 6 presents and discusses experimental results. Section 7 concludes and discusses future work.

2 Related Work

This paper investigates the problem of energy cost optimization and task resource scheduling in a dynamic cost environment for multiplex networked industrial chains task processing systems.

2.1 Energy cost optimization

Optimization of energy costs is the efficient use of energy to reduce energy utilization costs and improve energy efficiency within a system[3]. This optimization has numerous applications within the context of global energy shortage and rising energy costs. As optimization allocation has a broad application market, the optimization of energy costs has gained recognition as one of the active research areas[5–7], particularly in small energy utilization systems. The large-scale energy utilization systems, due to huge energy consumption, also present significant optimization space, making it a hot field of current research[3, 4, 8, 9].

The industrial energy system is a major component of large-scale energy systems. In the cost-optimized system for industrial energy, it is necessary to plan industrial task allocation, arrange the execution time and order of various industrial tasks reasonably, and improve energy utilization while reducing the energy cost of task execution. Zhou et al. studied energy cost optimization under the structure of multiplex microgrids. They studied and summarized energy allocation and control strategies in interactive energy trading, multi-task allocation, and flexible operation[3]. Schulze et al.[8] summarized the energy cost optimization of industrial systems and summarized the key factors and relevant conceptual framework of energy systems based on industrial subjects, indicating that energy cost optimization in industrial systems can effectively reduce energy use costs and improve production efficiency. Finally, Ullah et al.[4] proposed energy cost optimization strategies based on the locust optimization algorithm and cuckoo search optimization algorithm and verified the effectiveness of the algorithms.

Traditional task allocation and optimization methods under a simplex industrial chain make it difficult to solve the task allocation problem of multiplex networked industrial chains[10–12]. Therefore, it is necessary to develop new algorithms to process multiplex networked[13] industrial chains, which are complex networked shapes composed of multiplex different industrial chains interlocking, related, or overlapping with complex relationships[14] and interactions, including mutual impact characteristics of data, energy, and other aspects. This article focuses on the task processing system under multiplex networked industrial chains and needs to consider collaborative energy cost optimization in multiplex sub-systems. Suppose only the task allocation optimization technology in a simplex industrial chain is adopted in multiplex networked industrial chains. In that case, it is impossible to complete the optimization planning of cross-industry chain tasks, resulting in only achieving the best in a simplex industrial chain and being unable to improve energy utilization efficiency and reduce task execution costs in the overall system. It is recommended to explore new algorithms and consider multiplex networked industrial chains to solve this problem.

2.2 Task resource allocation

The problem of task resource allocation is a classic problem in the fields of computer science and operations research. This problem has a wide range of applications in real life, such as product agent production deployment planning[15, 16] and project schedule deployment arrangement[17–19].

The allocation of production workshop task resources is a classic problem that can be divided into single-machine scheduling problems, parallel machine scheduling problems, flow shop scheduling problems, and hybrid flow shop scheduling problems according to the properties of the task. In research in this area, Che et al.[20] considered a single-machine scheduling problem with a power-off mechanism. They established a mixed-integer linear programming model to find the optimal job processing sequence, thus minimizing the total energy consumption and maximum delay. Lei and Liu[21] proposed an artificial bee colony algorithm to introduce preventive maintenance constraints in the parallel machine scheduling problem, making it more in line with actual production needs. The algorithm's performance was extensively verified to minimize task completion time. In the flow shop scheduling problem, Zhao et al.[22] considered constraints such as distributed workshops and no idle time and proposed a self-learning Jaya algorithm to minimize tasks' total delay and energy consumption. Shao et al.[16] studied the distributed hybrid flow shop problem, combining the characteristics of distributed and parallel machine scheduling, and proposed a multi-neighborhood iterative greedy algorithm to solve the problem.

Resource-based deployment optimization considers the structure, scale, and usage restrictions of resources in the system and takes resource allocation as one of the main research objectives. The resource-constrained project scheduling problem is one of the classic problems in this field and has received much attention from scholars. Pellerin and others summarized and categorized heuristic algorithms in the resource-constrained project scheduling problem, indicating that current problem-solving algorithms are gradually transitioning from common to mixed heuristic algorithms. In addition, they compared the performance of different mixed heuristic algorithms on the project scheduling problem library (PSPLIB) dataset[23]. Rahman and others proposed a cultural genetic algorithm to solve the resource-constrained project scheduling problem and designed an automatic restart scheme to eliminate local optimal solutions[24]. Tirkolaee and others studied the multi-objective and multi-mode problems in resource-constrained project scheduling problems. They considered reusable resources, including labor, machines, equipment, and non-reusable resources, such as consumables, to make the model more realistic[25].

In summary, research in the traditional field of resource allocation for tasks has mainly focused on optimizing task execution time under various constraints. However, there has been less consideration of task allocation across systems and industry chains. These research results cannot fully apply to the multi-industry-chain environment with dynamic cost and time-of-use electricity pricing studied in this paper and cannot reflect the impact of multiplex networked industrial chains on task allocation.

3 Problem Formulation and Analyse

To clearly illustrate the research problem of this paper, we first introduce the relevant definitions of multiplex networked industrial chains in Section 3.1. We then introduce the description of the coarse time granularity allocation optimization problem of the multiplex industrial chain task processing system in Section 3.2. Finally, we give a formal formulation of the problem in Section 3.3.

3.1 Multiplex networked industrial chains

The task processing system of multiplex networked industrial chains is defined as Eq. (1):

$$G=\{m, wP, wS, mQ\} \tag{1}$$

The parameter m represents the composition of the system in terms of industrial chains, and the industrial Chains is denoted by Chain_i, $i\in\{1,2,\ldots,m\}$; wP is the power matrix of product agents machines in each industrial chain when performing tasks, wP_{jk}, $j\in\{1,2,\ldots,n\}, k\in\{1,2,\ldots,N\}$ represents the power of the product agents machine in the industrial chain where task J_{jk} is located when executing the task, where j is the task index number and k is the sub-task index number of task J. wS is the power matrix of the product agents machine in each industrial chain in the system when it is idle in standby, wS_{jk}, $j\in\{1,2,\ldots,n\}$, $k\in\{1,2,\ldots,N\}$ denotes the power of the machine in the industrial chain where task J_{jk} is idling; mQ denotes the matrix of the number of product agents in each industrial chain of the system, mQ_i, $i\in\{1,2,\ldots,m\}$ denotes the number of product agents in industry chain i.

The definition of tasks in multiplex industrial chains is as Eq. (2):

$$T=\{n, N, \chi, t_{\max}, pT\} \tag{2}$$

where n means that this task set consists of n independent tasks J_j, and task J_j consists of N subtasks, denoted as J_{jk}, $j\in\{1,2,\ldots,n\}$, $k\in\{1,2,\ldots,N\}$, χ represents the sequence of the industrial chain through which the subtasks are executed, $t_{\max}$

represents the deadline completion time of the batch of tasks, and all task allocation needs to be completed before the deadline, pT is the task execution time matrix, pT_{jk} represents the execution time required for task J_{jk}.

3.2 Coarse time granularity allocation optimization problem

Coarse time granularity allocation optimization problem of multiplex industrial chain task processing system (CTGAO) is defined as follows: Given task set T, multiplex networked industrial chains task processing system G, time-of-use electricity price $e(t)$, design an allocation strategy $\Pi < j,k,l,sT_jk,eT_jk >$ to optimize the time cost and energy cost of completing the batch of tasks.

In order to give a formal modeling of this problem, the following two decision variables are introduced first:

(1) For each subtask $J_{jk} \in J$, the binary decision variable r_{ijkl} indicates whether task J_{jk} is executed on the l-th product agent in industry chain i;

(2) In the case of time of use, electricity costs are calculated based on the amount of electricity consumed during task execution, considering the specific situation of tasks that span multiplex periods and their corresponding electricity consumption. As the number of tasks increases, the complexity of computing the cost of executing the task set increases drastically. In order to avoid the above problems, this paper introduces the binary decision variable y_{jkt}: For each subtask $J_{jk} \in J$, at any time $t \in [0, t_{\max}]$, the binary decision variable y_{jkt} indicates whether task J_{jk} is executing at time t. At the same time, t is assumed to be an indivisible minimum time unit, and the start and end of task execution and the change of electricity price must occur at the boundary of the time unit.

3.3 Problem formulation

Through the above definition of decision variables, the CTGAO can be modeled as a mixed integer linear programming model with the following form of Formulas (3)–(11). Among them, Formula (3) states that for subtasks of the same task, $J_{j(k+1)}$ needs to be completed after J_{jk} before execution can start. Equation (4) states that for any subtask J_{jk}, there can be only one product agent for execution. Formulas (5)–(7) ensure that any component can execute at most one task at a time, where $Z_{jkj'k'}$ means that task J_{jk} is executed before $J_{j'k'}$, and M is an integer large enough to ensure the correctness of the constraint. Equations (8) and (9) ensure that the end time and execution time of the task meet the requirements. Formula (10) defines the maximum completion time. Equation (11) describes the power cost calculation method for task execution.

$$\begin{aligned} &\min(\text{energyCost}, C_{\max}) \qquad (\text{CTGAO}) \\ \text{s.t.}\quad & sT_{jk} + pT_{jk} \leqslant T_{j(k+1)} \end{aligned} \tag{3}$$

$$\sum_{l=1}^{mQ_i} r_{ijkl} = 1, i \in \{1,2,\ldots,m\} \tag{4}$$

$$\begin{aligned} &M\left(2 - r_{ijkl} - r_{ij'k'l}\right) + M\left(1 - Z_{jkj'k'}\right) + \\ &sT_{j'k'} - sT_{jk} \geqslant pT_{jk}, \\ &j < j', j, j' \in \{1,2,\ldots,n\}, \\ &k \in \{1,2,\ldots,N\}, l \in \{1,2,\ldots,mQ_i\} \end{aligned} \tag{5}$$

$$\begin{aligned} &M\left(2 - r_{ijkl} - r_{ij'k'l}\right) + M \times Z_{jkj'k'} + \\ &sT_{jk} - sT_{j'k'} \geqslant pT_{j'k'} \end{aligned} \tag{6}$$

$$Z_{jkjk} = 0 \tag{7}$$

$$eT_{jk} = sT_{jk} + pT_{jk} - 1 \tag{8}$$

$$pT_{jk} = \sum_{t=1}^{t_{\max}} y_{jkt} \tag{9}$$

$$eT_{jk} \leqslant C_{\max} \tag{10}$$

$$\begin{aligned} &\text{energyCost} = \sum_{j=1}^{n} \sum_{k=1}^{N} \sum_{t=1}^{t_{\max}} \\ &\left(y_{jkt} * e(t) * wP_{jk} + \left(1 - y_{jkt}\right) * wS_{jk}\right) \end{aligned} \tag{11}$$

4 Branch and Bound Method in Small-Scale Scenario

The CTGAO problem is modeled as a bi-objective optimization problem under mixed integer linear programming (MILP). In the optimization problem with more than one objective, it is generally impossible to obtain an optimal solution such that all objective function values are superior to any other solution. In order to solve this problem, this paper uses the linear weighted sum method[26], transforms the optimization objective into a single objective, and then proposes an exact algorithm based on the branch and bound method to solve the problem. By using the linear weighted sum method, the optimization objective is transformed into Eq. (12):

$$\min f = \alpha C_{\max} + (1-\alpha)\text{energyCost} \tag{12}$$

4.1 Algorithm idea

Based on the branch and bound strategy, this section proposes an exact algorithm for solving the CTGAO problem in small-scale scenarios. The basic idea of the branch and bound method is to repeatedly divide the feasible solution space into smaller and smaller subsets and compute the objective bounds for each subset. After each branch, the nodes that exceed the known objective bounds of the feasible solution space are pruned. By optimizing the search space through pruning, the size of the solution space is reduced until the entire search space is thoroughly explored[27]. Based on this concept, this paper proposes an optimal algorithm called BranchBoundOPT for small-scale scenarios.

Firstly, the solution space of the problem is constructed. For the CTGAO problem, all 0-1 decision variables are first constructed as a node in the tree, each level represents a 0-1 variable, and the tree formed by all nodes is the solution space. In the solution set tree, a path from the root node to a leaf node is a solution, and the solution space construction is shown in Fig. 1.

This paper implements a fast pruning operation on the solution space during the initialization phase in order to increase the solving speed and reduce the size of the solution space. This operation is based on the characteristics of the task processing system in multiplexed networked industrial chains. Its purpose is to reduce the solution space size of the subsequent search. The specific procedure is as follows:

(1) Chain-level fast pruning: if r_{ijkl} is 1, then all nodes whose $r_{i'jkl'}$ is 1 are pruned, where $i \neq i'$, $j \neq j'$;

(2) Task-execution-level fast pruning: if y_{jkt} is 1, then all nodes whose $y_{jkt'}$ is 1 are pruned, where $t' > t + pT_{jk}$ or $t' < t - pT_{jk}$.

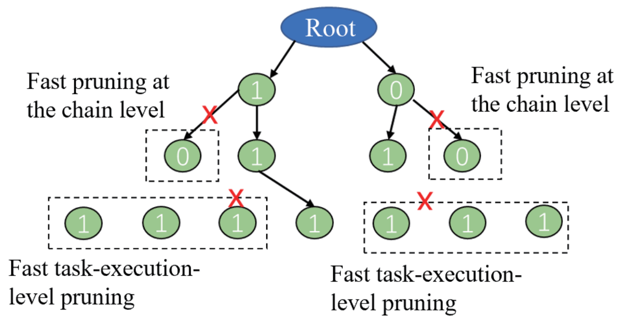

Root
Fast pruning at the chain level
Fast pruning at the chain level
Fast task-execution-level pruning
Fast task-execution-level pruning

Fig. 1 Example of solution space initialization of branch and bound method.

4.2 Algorithm description

According to the algorithm idea in Section 4.1, the detailed description of the BranchBoundOPT algorithm based on the depth-first branch and bound method[27] is given below:

As shown in Algorithm 1, the Initialized Solution algorithm generates the first solution space tree, and the original problem is relaxed into a linear relaxation problem (LRP) problem. The solution of the LRP problem is calculated to obtain the lower bound of the original problem solution, and then the upper bound is initialized to a sufficiently large integer. The solute on space is then searched recursively using the BranchSearch algorithm, which keeps pruning, updating the upper bound, and updating the lower bound until the solution space tree traversal ends. The task set J, the multiplex networked industrial chains task processing system G, and the time-sharing electricity price $e(t)$ contain all the information needed to calculate the allocation task, lower-bound and upper-bound are global variables, and the current lower bound and upper bound of the optimal solution of the task are recorded respectively.

The initialize solution algorithm generates a solution space tree according to the number of tasks in the task set and the composition of the system's components and calculates and updates the initial lower bound. Where num is the number of 0-1 decision variables, the queue is a first in, first out (FIFO) queue container, and quick-cut is a fast-pruning strategy. According to the characteristics of the task processing system of multiplex networked industrial chains, infeasible nodes are deleted when constructing the initial solution space, which can greatly reduce the solution space. The specific algorithm is given in Algorithm 2.

Algorithm 1 Overall flow of the optimal algorithm BranchBoundOPT

Input: Task set J, multiplex networked industrial chain task processing system G, time-of-use electricity price $e(t)$;
Output: Task set optimal allocation plan bestSolution;

```
1 Initialize: lowerBound ← 0, upperBound ← maxInteger;
2 soultionTree ← initializeSolution(J, G);
3 Call the LP solver to find the LRP solution of root;
4 updatelowerBound;
5 bestSolution ← BrandSearch(solutionTree, lowerBound, upperBound);
6 return bestSolution;
```

Algorithm 2 Initialize Solution

```
Input: Task set $J$, multiplex networked industrial chains task processing system $G$, time-of-use electricity price $e(t)$;
Output: Task set solution space soultionTree;
1 Initialize: num ← 0, root.val ← 0, queue ← null;
2 Calculate num according to Formulas (3)–(12);
3 queue.add(root);
4 for $i$ ← to num do
5     size ← queue.size();
6     for $j$ in 1 to size do
7         node.poll();
8         node.left.val ← 0;node.right.val ← 1;
9         if quickCut(node.left) then
10            queue.add(node.left);
11        if quickCut(node.right) then
12            queue.add(node.right);
13  soultionTree ← root;
14 return soultionTree;
```

BranchSearch searches the solution space recursively, starts from the root node of the solution space tree, and updates the upper and lower bounds continuously during the search process. It optimizes the search space by pruning to improve the search efficiency. stack is a first-in, last-out stack container, which can ensure that the solution space is searched according to depth-first. The specific algorithm is shown in Algorithm 3.

Algorithm 3 Initialize Solution

```
Input: Solution space solutionTree, lowerBound, upperBound;
Output: Optimal allocation scheme of task set bestSolution;
1 Initialize: stack ← null;
2 stack.add(soultionTree);
3 while stack is not empty do
4     node ← stack.pollLast();
5     res←Calculate the optimal solution of LRP problem
6     if res is not null then
7         if node is leaf then
8             if res¡upperBound then
9                 upperBound ← res
10            else
11                continue
12        else
13            if resupperBound then
14                lowerBound ← res;
15                stack.add(node.left);
16                stack.add(node.right);
17            else
18                continue
19 bestSolution ← upperBound;
20 return bestSolution;
```

5 Metaheuristic Cascade Adaptive Algorithm in Large-Scale Scenarios

The algorithm proposed in the previous section of this paper results in exponential growth of solution time when there are increases in either the number of tasks or the number of industrial chains in the problem. This makes it difficult to meet the needs of large-scale problem scenarios. To address this issue, this section proposes a meta-heuristic cascading adaptive algorithm suitable for large-scale scenarios that can obtain approximate optimal solutions within an acceptable time frame.

5.1 Algorithm idea

Previous studies have proven that the production workshop scheduling problem is an NP-hard problem[16, 28]. In this paper, we study the production workshop scheduling problem under the background of multiplex networked industrial chains, which further increases the complexity of the problem. In large-scale problems, the cost of obtaining the optimal solution is high. Based on previous heuristic evolutionary algorithm research[29], we propose a cascaded adaptive algorithm (CAA) based on fitness rank sorting. The algorithm proposed in this paper adopts the meta-heuristic algorithm process and designs corresponding adaptive strategy modification mechanisms in response to the characteristics of multiplex networked industrial chains, as follows:

Previous metaheuristic algorithms usually require variations of the solution set to increase diversity. However, in this paper, direct variations and crossovers of the solution set may cause the loss or repetition of subtasks in new solutions or may not satisfy the constraint conditions. Therefore, this paper modifies and evolves the solution set through neighborhood insertion and exchange. The characteristics of the multiplex networked industrial chain make it so that when using neighborhood insertion and exchange, subtasks with later industrial chain sequences will be moved to the front, resulting in invalid solutions. Therefore, this paper designs a cascading adaptive strategy to adaptively adjust the order of subtasks of the same task to avoid generating invalid solutions.

5.2 Algorithm description

The detailed design of each step of the CAA algorithm is given below:

Step 1: Generate an initial set of solutions.

Randomly initialize the execution order of tasks and the selection of product agents. Then, according to the execution order of the tasks, initialize the start and end times of each task one by one. The start time of a task is the later of the earliest available time of its product agents and the earliest start time of the task, as follows:

$$sT_{ij} = \max\left(\text{availableTime}_{il}, eT_{j(k-1)}\right) \tag{13}$$

At the same time, update the earliest available time of the corresponding product agents and set the earliest available time of the task's next subtask to be the current subtask's end time. Initialize all tasks to generate a feasible solution. Repeat the above steps to generate the initial solution set.

Step 2: Neighborhood insertion and cascading adaptive transmission.

Aiming at the feasible solution A, a task Job_{jk} is randomly selected from the solution, and Job_{jk} is inserted into the gap of other tasks in turn to obtain a new solution. Since tasks have inter-chain cascading relationships, the insertion of this task into the gap of other tasks may bring infeasible solutions. Therefore, this paper refers to the previous literature[29] and designs a cascade adaptive conduction mechanism: When a subtask is inserted after the successor task of the task, the subtask execution order of the task is automatically updated. For example, suppose the original task execution order is $\text{Job}_{21}-\text{Job}_{11}-\text{Job}_{22}-\text{Job}_{12}-\text{Job}_{23}$. inserting Job_{21} into other gaps yields new solutions, one of which is $\text{Job}_{11}-\text{Job}_{22}-\text{Job}_{12}-\text{Job}_{21}-\text{Job}_{23}$. In this case, since task Job_{21} is executed after task Job_{22} and the constraint condition is not satisfied, the cascade adaptive transmission is executed, and the solution is readjusted as $\text{Job}_{11}-\text{Job}_{21}-\text{Job}_{12}-\text{Job}_{22}-\text{Job}_{23}$.

Step 3: Random neighborhood exchange and cascaded adaptive conduction.

The random neighborhood exchange and cascading adaptive propagation based on the multiplex networks industrial chain aims to improve the quality of the solution. The method randomly selects two feasible solutions, A and B, exchange part of the task sequence fragments between A and B, and then cascade and adaptively propagate the modified solution after the exchange, resulting in a new feasible solution. However, it should be noted that neighborhood exchange may cause inconsistencies between the exchanged task segments, resulting in lost or redundant tasks in the solution. To solve this problem, we convert the redundant tasks into the missing tasks in the sequence from front to back. We repeat the above steps until all solutions in the set have completed the random neighborhood exchange.

Step 4: Merge the above solutions with the original solution set, and evaluate the generated intermediate solution set.

Calculate the objective function values of all solutions in the intermediate solution set. Suppose solution A's objective function values are superior to solution B's. In that case, it indicates that Solution B is dominated by Solution A. Calculating the number of times each solution is overlooked and sort the solution set by the degree of dominance. If two solutions have the same degree of dominance, sort these two solutions by their crowding distance. The formula for calculating the crowding distance is as follows:

$$\text{CrowdingDistance}_i = \sum_{m=1}^{M} \frac{f_m(i+1) - f_m(i-1)}{(f_m^{\max} - f_m^{\min})} \tag{14}$$

where M represents the number of objective functions, $M = 2$ in this paper, $f_m(i+1)$ and $f_m(i-1)$ are the objective function values of the two solutions closest to the objective function value of solution i in the solution set, $f_m^{\max}$ and $f_m^{\min}$ are the maximum value and minimum value of the m-th objective function, respectively. The best S solutions after sorting are screened out as the feasible solution set of the next iteration, where S is the solution set size.

Step 5: Repeat the above process until the predetermined number of iterations is reached, and then output the current solution set. The pseudocode of the overall algorithm is shown in Algorithm 4.

6 Experiment

In this section, a series of experiments are carried out to test the proposed algorithms. The effectiveness and efficiency of our proposed algorithms are verified by comparison with the classical benchmark.

6.1 Experimental settings

The experiments were run on a PC with a 3.6 GHz central processing unit, Windows 10 operating system, and 8 GB of working memory. The test code was coded using Matlab R2019b, the LRP problem was

Algorithm 4 Overall process of CAA algorithm

```
Input: Solution set size $S$, number of iterations Iterations;
Output: Pareto optimal solution set;
1 Initialize: $T, G, e(t)$, objective // Initialize the task information;
2 solutions ← initializeSolution(); // Generate the initial set of solutions
3 for $i = 1$ to Iterations do
4     for solution in solutions do
5         midSolution1 ← Neighborhood insertion and adaptive
6         midSolution2 ← Neighborhood insertion and adaptive
7     midSolution ← merge(solutions, midSolution1, midSolution2)
8     sort(midSolution)
9     solutions ← replace(midSolution)
10 return bestSolution;
```

solved by calling CPLEX software, and each set of experiments was repeated 20 times for the average.

In the simulation experiment, the task allocation and task allocation optimization scenarios of the multiplex industrial chain task processing system are simulated. When the garbage disposal task arrives, the task execution sequence and the execution product agent need to be calculated, and the allocation scheme is formed.

The number of network layers in the model is represented by m; The number of tasks is represented by n; the execution time required for each subtask is denoted by pT_{ij}; the number of components in each network layer is represented by machineQty_i; the execution unit time power of members in each network layer is expressed in wP_i; the part idle unit time power wS_i in each network layer. The time-of-use price is shown in Eq. (15):

$$e(t)=\begin{cases}0.3551, & 0\leqslant t\leqslant 8;\\ 1.2757, & 8\leqslant t\leqslant 12, 17\leqslant t\leqslant 21;\\ 0.7653, & 12\leqslant t\leqslant 17, 21\leqslant t\leqslant 24\end{cases} \tag{15}$$

6.2 Benchmark algorithms

In order to verify the feasibility of the algorithm in this paper and the limitations of the previous task deployment optimization methods under a simplex chain in multiplex chains, the following algorithm was selected as a comparison algorithm for analysis.

Simplex network Heuristic iterative greedy allocation algorithm (SNHA): The heuristic iterative greedy algorithm is widely used in the shop allocation problem. Therefore, this paper considers the iterative greedy strategy under the structure of a simplex industrial chain proposed in previous studies to be hierarchically applied to the scene of multiplex industrial chains for allocation calculation[30].

Improved fast non-dominated sorting genetic algorithm-II (NSGA-II) algorithm is widely used in multi-objective optimization solutions. In this paper, referring to previous cloud computing deployment, the algorithm is improved and applied to multiplex industrial chain scenarios, the solution set size is set to 50, and the number of iterations is set to 200[31].

Parallel scheduling optimization algorithm (PSO): In this paper, the previous parallel batch processing optimization algorithm is applied in the scenario of multiplex industrial chains, and parallel optimization is carried out under the structure of multiplex networked industrial chains[32].

6.3 Experiments with exact algorithms in small-scale scenarios

The experimental verification will be in small-scale scenarios for the proposal of an exact algorithm. This section considers six indicators [0.0, 0.2, 0.4, 0.6, 0.8, 1.0] for the bi-objective weight to obtain more accurate results. First, the task size was kept constant, and the maximum completion time of the task execution and the energy cost of the task execution were tested under different weights. Then the running time of the algorithm, the maximum completion time and the energy cost of the algorithm are given.

6.3.1 Influence of different weights

Parameter settings: The exact algorithm for the small-scale scenario proposed in this paper in Section 4 needs to convert the double objectives into a single objective by the linear weighted sum method, so this section tests the algorithm running results with different weights. The fixed problem size is 3 × 4 × 3 problems with 3 industrial chains and 4 tasks, and each task has 3 processes. Due to the large difference between the value of the makespan $C_{\max}$ and the value of the electricity cost energyCost, energyCost is magnified 10 times to be processed when calculating the result. The experimental results are shown in Fig. 2.

Phenomenon & reason: As can be seen from Fig. 2, as the value of α increases, the maximum completion time of tasks decreases, but the energy cost required to complete the task increases. These two indicators are negatively correlated. Striving for fast task completion will increase energy cost, while striving for lower

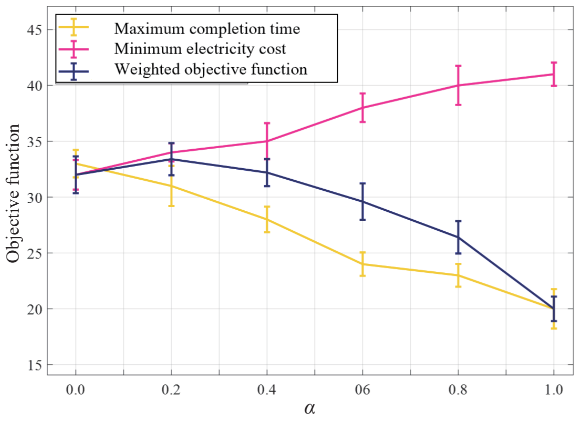


Fig. 2 Experimental results for different values of *α*.

energy cost will delay task completion time. In real task environments, product agents need to choose the appropriate α value based on specific circumstances. In other experiments in this paper, we chose $\alpha = 0.5$ for ease of computation.

6.3.2 Influence of task quantity variation, average subtask quantity variation, and network layers

Parameter settings: The specific experimental results are shown in Figs. 3a and 3b for a fixed network layer number of 3 and an average number of sub-tasks of 4. Figs. 3c and 3d illustrates the impact of subtask quantity variation on algorithmic operation results with a fixed task quantity of 4 and network layer quantity of 3. The task number is set at 4, with an average of 4 subtasks per task. The properties of each network layer are randomly generated within their respective ranges, and the averaged experimental results are shown in Figs. 3e and 3f.

Phenomenon & reason: The operation results of the algorithm in the small-scale scenario are shown in Fig. 3. Figure 3a shows the variation of max completion time for different algorithms when the problem size changes, while Fig. 3b shows the variation of electricity cost when the problem size changes. The proposed BranchBoundOPT algorithm outperforms other comparative algorithms in terms of maximum task completion time and power consumption at different problem scales. Figure 3c shows the impact of the average number of subtasks on the longest completion time of a task. As the average number of subtasks increases, the maximum completion time of each algorithm's calculated task also increases. Figure 3d shows the impact of average subtask size on power consumption during task

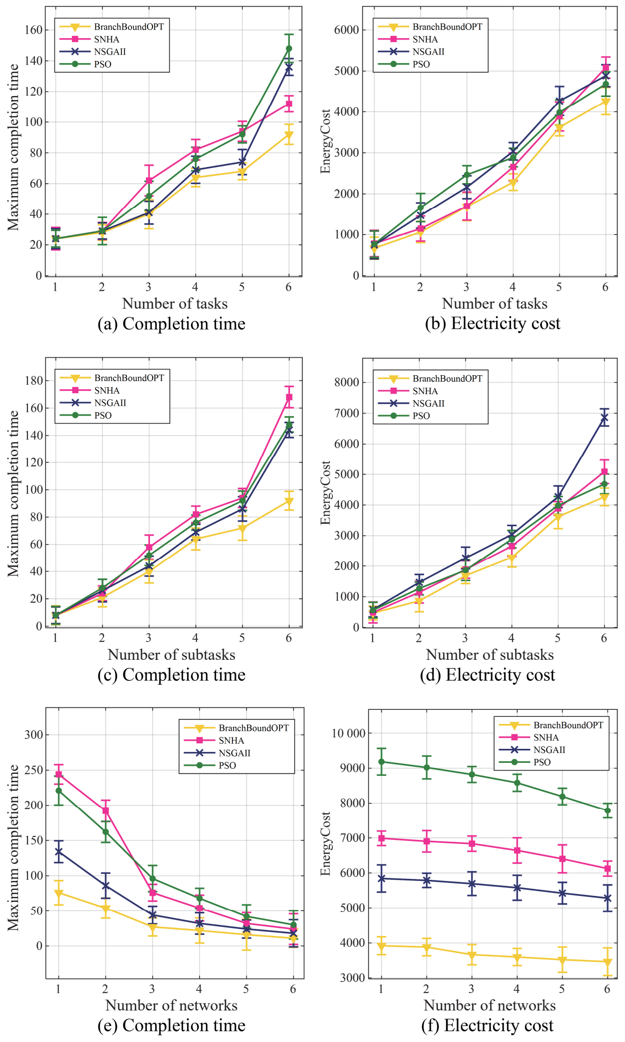


Fig. 3 Results of the algorithm in a small-scale scenario.

execution. As the average number of subtasks increases, the power consumption required to execute tasks in each algorithm also increases. Figure 3e shows the impact of network layers on the longest task completion time. In all cases, the longest completion time of the BranchBoundOPT algorithm is smaller than the shortest completion time of the comparison algorithm. Figure 3f shows the impact of the network layer on the power cost of task execution. As the number of network layers increases, the power consumption cost of tasks solved by various algorithms will decrease, but the speed of reduction is relatively slow. The PSO algorithm has the most obvious downward trend, while the BranchBoundOPT algorithm has the smoothest downward trend.

6.3.3 Influence of task size variations on algorithm running time

Phenomenon & reason: Figure 4 shows the number of

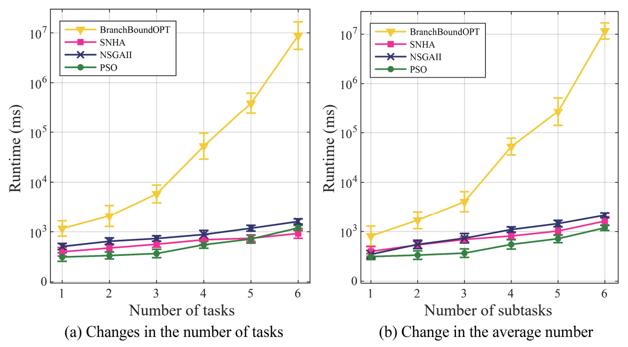


Fig. 4 Influence of task size variations on algorithm execution time.

tasks and the average number of tasks increase, the problem scale also grows. Therefore, in the BranchBoundOPT algorithm, the solution space of the precise method exponentially increases with the problem scale, resulting in an exponential increase in solution time. On the other hand, the different compared algorithms use heuristic solutions, so their runtime increases linearly with the problem scale, although they are slower than the precise algorithm. When the problem scale remains unchanged, the change in the number of network layers does not directly lead to an increase in algorithm computational complexity, thus keeping the algorithm's runtime stable. Therefore, the experimental results suggest that the precise algorithm is unsuitable for larger problem scenarios.

6.4 Metaheuristic cascade adaptive algorithms in large-scale scenarios

This subsection presents the experimental validation of the proposed heuristic cascade adaptive algorithm in large-scale scenarios. Since the algorithm is a multi-objective optimization algorithm, various basic indicators are tested in addition to the algorithm's running time. These include the maximum completion time of task execution, power cost for task execution, and other metrics. The Pareto solution set generated by the algorithm is also compared with the SNHA, NSGAII, and PSO algorithms used in Section 2.6.2 as the benchmark comparison algorithms. The large-scale problem scenario is based on the test problem described in reference[33], which is further improved for this example. The specific evaluation index is as follows:

Non-dominant indicator (NI)[31]: It represents the proportion of non-dominant solutions in the solution set obtained by the algorithm among all solutions obtained by the algorithm. As shown in Eq. (16):

$$\mathrm{NI} = \frac{|\pi_a | \pi_a \in D \cup \pi_a \in D^* \,|}{|D^*|} \tag{16}$$

Distribution indicator (DI)[34]: Used to evaluate the distribution of the Pareto solution set obtained by the algorithm. As shown in Eq. (17):

$$\mathrm{DI} = \frac{d_f + d_l + \sum_{i=1}^{N-1} \left| d_i - \bar{d} \right|}{d_f + d_l + (F-1)\bar{d}} \tag{17}$$

Inverted generational distance (IGD)[35]: The convergence of an algorithm and the diversity of non-inferior solutions are evaluated by calculating the sum of the nearest distances between the points on the optimal Pareto front and the Pareto front obtained by the algorithm. As shown in Eq. (18):

$$D_{\mathrm{IGD}} = \frac{\sum_{x \in N^*} d(x, N)}{|N^*|} \tag{18}$$

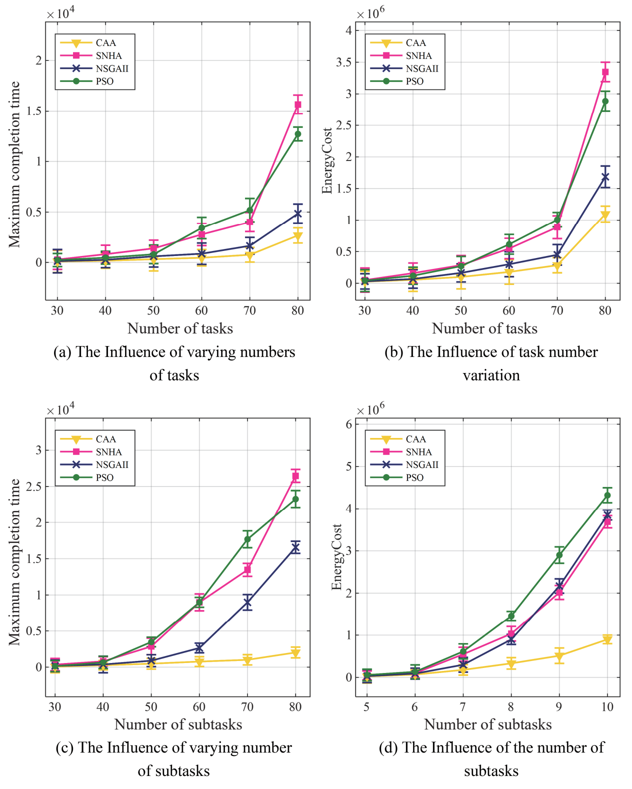
6.4.1 Influence of task size variations on algorithm running time

Phenomenon & reason: The influence of variations in task size on the algorithm performance in large-scale scenarios is shown in Fig. 5. First, in the case of 5 layers of networked, we fixed the average number of subtasks per task at 7 and showed the effect of changing the number of tasks on the maximum completion time and minimum energy cost, as shown in Figs. 5a and 5b. Then we fixed the number of tasks at 60 and showed the effect of the average number of subtasks on the maximum completion time and minimum power cost, as shown in Figures 5c and 5d. As the size of the problem increases, the maximum completion time and energy cost of the task continue to increase. The quality of the solutions proposed by the CAA algorithm of this paper is better than that of other comparative algorithms.

Although the CAA algorithm proposed in this paper is slower in running time compared to the SNHA algorithm, the quality of the solution is significantly better. As the scale of the problem increases, the difference between the algorithm results becomes even greater, indicating that the CAA algorithm proposed in this paper can optimize the task execution time and cost in large-scale problems.

6.4.2 Comparison of Pareto solution sets of algorithms

Phenomenon & reason: We take the 5 × 50 × 5 task scale as an example, and solve the Pareto solution set



(a) The Influence of varying numbers of tasks

(b) The Influence of task number variation

(c) The Influence of varying number of subtasks

(d) The Influence of the number of subtasks

Fig. 5 Influence of task size variations on algorithm execution time.

and analyze the results of the comparison. The specific experimental results are shown in Fig. 6. It is apparent that the heuristic cascade adaptive algorithm designed in this paper can achieve relatively excellent comprehensive solutions when the test examples are the same. Moreover, it outperforms other comparative algorithms in most cases. First, in terms of maximum completion time of tasks, the Pareto optimal solution obtained by the algorithm proposed in this paper is

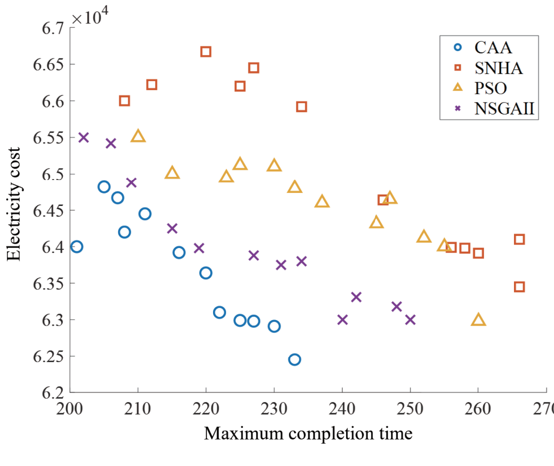


Fig. 6 Comparison of Pareto solution sets for 5 × 50 × 5 problems.

obviously better than that of the comparative algorithm. Secondly, the proposed algorithm achieves lower energy cost in most cases, indicating that the CAA algorithm can obtain relatively excellent solutions in the context of large-scale problems.

6.4.3 Evaluation of the solution set of the algorithm

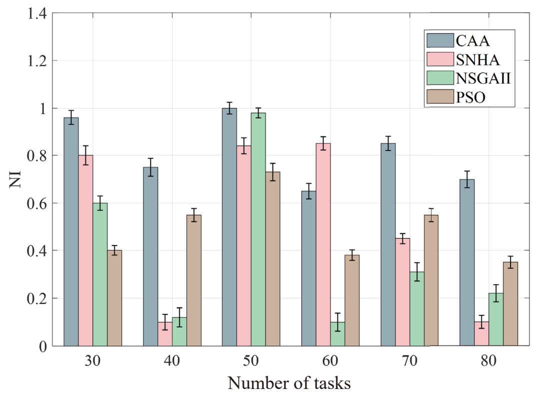
Parameter settings: We selected 5 industry chains and maintained an average of 7 subtasks while investigating the effect of the number of tasks on the solution set. In addition, the number of tasks was set to 60 to assess the influence of the number of subtasks on the solution set.

Phenomenon & reason: The results are shown in Fig. 7. The figure clearly shows that the metrics NI, DI, and IGD achieved by the proposed CAA algorithm exceed those of other algorithms in most cases. This indicates that the CAA algorithm can obtain higher quality solutions when addressing the challenges posed by large-scale, multiplex industrial chain task processing system problems. In addition, the resulting solution set exhibits improved distribution and convergence characteristics.

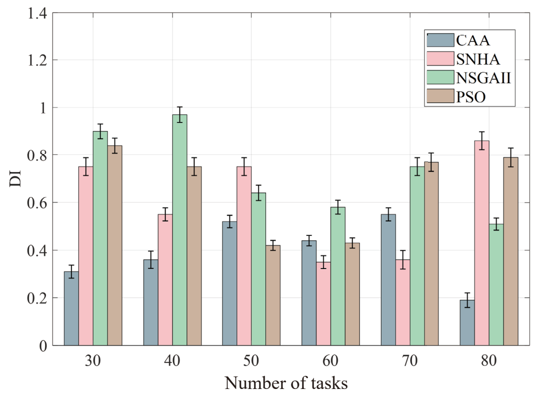
The proposed BranchBoundOPT algorithm can effectively optimize resource scheduling problems in the context of multiple networked industrial chains, reducing completion time and power consumption. However, the required computational time of the algorithm increases rapidly with the size of the problem. To address problem solving in large-scale scenarios, we propose the CAA algorithm, which results in a better distribution and convergence of the solution set. The experiment shows that both algorithms are superior to the comparison algorithm for scenarios of different scales.

7 Conclusion

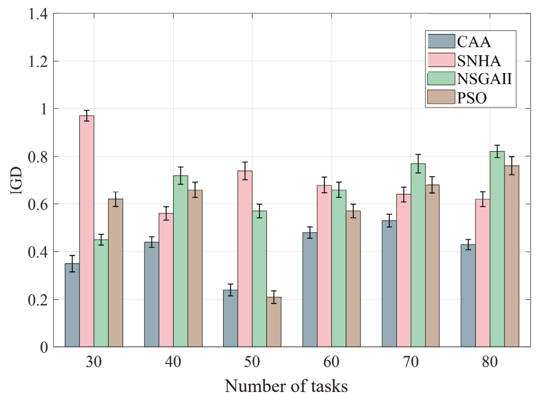
This paper focuses on time-of-use price through coarse time granularity task schedules in multiplex networked industrial chains. It addresses the limitations of traditional task allocation optimization methods designed for simplex industrial chains and presents a specialized task allocation optimization model for multiplex networked industrial chains. For small-scale problems, an efficient exact algorithm based on the branch-and-bound method with a fast-pruning strategy is proposed. This algorithm provides optimal allocation results promptly. Experimental results demonstrate its success in small-scale scenarios. However, considering



（a）NI values of the solution set for different problem sizes



（b）DI values of the solution set for different problem sizes



（c）IGD values of solution sets for different problem sizes

Fig. 7 Solution set evaluation parameters for different problem sizes.

the computational complexity, the exact algorithm is unsuitable for large-scale problems. Hence, a heuristic cascading adaptive algorithm is introduced to obtain approximate optimal solutions within a reasonable time frame for large-scale problems. The experimental results indicate that the proposed method effectively optimizes task execution time and energy cost in the task processing system of the multi-networked industrial chain with coarse time granularity. Comparisons with the direct application of traditional algorithms demonstrate the superiority of the proposed method. The resource allocation optimization problem of the multiplex networked industrial chains system studied in this article mainly considers scenarios where online tasks with different priorities dynamically arrive. In actual industrial chain power allocation systems, there may also be scenarios such as system task execution component failures and task execution failures, which will have a significant impact on the current allocation. Therefore, it is necessary to analyze this scenario, and design corresponding algorithms to handle the dynamics of the system based on the corresponding characteristics.

Acknowledgment

This work was supported by the National Key Research and Development Program of China (No. 2022YFB3304400), the National Natural Science Foundation of China (Nos. 62303111, 62076060, and 61932007), the Key Research and Development Program of Jiangsu Province of China (No. BE2022157), the Defense Industrial Technology Development Program (No. JCKY2021214B002), and the Fellowship of China Postdoctoral Science Foundation (No. 2022M720715).

References

[1] P. Li, K. Di and X. Bai, Fine Time Granularity Allocation Optimization of Multiple Networks Industrial Chains in Task Processing Systems, presented at the 24th International Conference on Parallel and Distributed Computing, Applications and Technologies, Jeju, Korea, 2023.

[2] C. Potts and M. Kovalyov, Scheduling with batching: A review, *European Journal of Operational Research*, vol. 120, no. 2, pp. 228–249, 2000.

[3] B. Zhou, J. Zou and C. Chung, et al, Multi-microgrid energy management systems: Architecture, communication, and scheduling strategies, *Journal of Modern Power Systems and Clean Energy*, vol. 9, no. 3, pp. 463–476, 2021.

[4] I. Ullah, I. Hussain and M. Singh, Exploiting grasshopper and cuckoo search bio-inspired optimization algorithms for industrial energy management system: Smart industries, *Electronics*, vol. 9, no. 1, pp. 105–120, 2020.

[5] B. Zhou, W. Li and K. Chan, et al, Smart home energy management systems: Concept, configurations, and scheduling strategies, *Renewable and Sustainable Energy Reviews*, vol. 61, pp. 30–40, 2016.

[6] H. Rocha, I. Honorato and R. Fiorotti, et al, An Artificial Intelligence based scheduling algorithm for demand-side energy management in Smart Homes, *Applied Energy*, vol. 282, pp. 116145–116162, 2021.

[7] S. Aman, Y. Simmhan, and V. Prasanna, Energy management systems: state of the art and emerging trends, *IEEE Communications Magazine*, vol. 51, no. 1, pp. 114–119, 2013.

[8] M. Schulze, H. Nehler and M. Ottosson, et al, Energy management in industry–a systematic review of previous findings and an integrative conceptual framework, *Journal of cleaner production*, vol. 112, pp. 3692–3708, 2016.

[9] D. Wu, G. Zeng and D. He, et al, Task Coordination Organization Model and the Task Allocation Algorithm for Resource Contention of the Syncretic System, *Tsinghua Science and Technology*, vol. 21, no. 4, pp. 459–470, 2016.

[10] F. Han and K. Di, Multi-robot Task Allocation in the Environment with Functional Tasks, presented at the 31st International Joint Conference on Artificial Intelligence, Vienna, Austria, 2022.

[11] Y. Jiang, Y. Zhou, Y. and Li, Reliable task allocation with load balancing in multiplex networked, *ACM Transactions on Autonomous and Adaptive Systems* (*TAAS*), vol. 10, no. 1, pp. 1–32, 2015.

[12] Z. Zhao, M. Zhou, and S. Liu, Iterated greedy algorithms for flow-shop scheduling problems: A tutorial, *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 3, pp. 1941–1959, 2021.

[13] Z. Li, F. Yan, and Y. Jiang, Cross-layers cascade in multiplex networked, *Autonomous Agents and Multi-Agent Systems*, vol. 29, pp. 1186–1215, 2015.

[14] K. Li, S. Wu and Y. Wen, et al, Task Allocation of Multiagent Groups in Social Networked Systems, *IEEE Internet of Things Journal*, vol. 9, no. 14, pp. 12194–12208, 2021.

[15] Y. Pan, Q. Ran and Y. Zeng, et al, Symmetric Bayesian Personalized Ranking With Softmax Weight, *IEEE Transactions on Systems, Man.and Cybernetics*: *Systems*, vol. 53, no. 7, pp. 4314–4323, 2023.

[16] W. Shao, Z. Shao, and D. Pi, Modeling and multi-neighborhood iterated greedy algorithm for distributed hybrid flow shop scheduling problem, *Knowledge-Based Systems*, vol. 194, pp. 105527–105544, 2020.

[17] S. Hartmann and D. Briskorn, An updated survey of variants and extensions of the resource-constrained project scheduling problem, *European Journal of Operational Research*, vol. 297, no. 1, pp. 1–14, 2022.

[18] R. Pellerin, N. Perrier, and F. Berthaut, A survey of hybrid metaheuristics for the resource-constrained project scheduling problem, *European Journal of Operational Research*, vol. 280, no. 2, pp. 395–416, 2020.

[19] F. Han and K. Di, Solving the Multi-robot task allocation with functional tasks based on a hyper-heuristic algorithm, *Applied Soft Computing*, vol. 14, no. 110628, 2023.

[20] A. Che, X. Wu and J. Peng, et al, Energy-efficient bi-objective single-machine scheduling with power-down mechanism, *Computers & Operations Research*, vol. 85, pp. 172–183, 2017.

[21] D. Lei and M. Liu, An artificial bee colony with division for distributed unrelated parallel machine scheduling with preventive maintenance, *Computers & Industrial Engineering*, vol. 141, pp. 106320–106331, 2020.

[22] F. Zhao, R. Ma and L. Wang, A self-learning discrete jaya algorithm for multiobjective energy-efficient distributed no-idle flow-shop scheduling problem in heterogeneous factory system, *IEEE Transactions on Cybernetics*, vol. 52, no. 12, pp. 12675–12686, 2021.

[23] R. Pellerin, N. Perrier, and F. Berthaut, A survey of hybrid metaheuristics for the resource-constrained project scheduling problem, *European Journal of Operational Research*, vol. 280, no. 2, pp. 395–416, 2020.

[24] H. Rahman, R. Chakrabortty, and M. Ryan, Memetic algorithm for solving resource constrained project scheduling problems, *Automation in Construction*, vol. 111, pp. 103052–103070, 2020.

[25] E. Tirkolaee, A. Goli and M. Hematian, et al, Multi-objective multi-mode resource constrained project scheduling problem using Pareto-based algorithms, *Computing*, vol. 101, pp. 547–570, 2019.

[26] G. Fan, J. Wang, and Z. Liu, Two-agent scheduling on mixed batch machines to minimise the total weighted makespan, *International Journal of Production Research*, vol. 61, no. 1, pp. 238–257, 2023.

[27] E. Lawler, and D. Wood, Branch-and-bound methods: A survey, *Operations research*, vol. 14, no. 4, pp. 699–719, 1966.

[28] C. Lu, L. Gao and X. Li, et al, Energy-efficient permutation flow shop scheduling problem using a hybrid multi-objective backtracking search algorithm, *Journal of Cleaner Production*, vol. 144, pp. 228–238, 2017.

[29] C. Liu, Z. Xu and Q. Zhang, et al, Green scheduling of flexible job shops based on NSGA-II under TOU power price, *China Mechanical Engineering*, vol. 31, no. 05, pp. 576–585, 2020.

[30] Z. Zhao, M. Zhou, and S. Liu, Iterated greedy algorithms for flow-shop scheduling problems: A tutorial, *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 3, pp. 1941–1959, 2021.

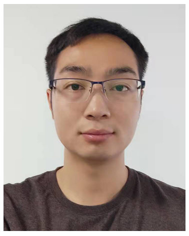
[31] H. Li, B. Wang and Y. Yuan, et al, Scoring and dynamic hierarchy-based NSGA-II for multiobjective workflow scheduling in the cloud, *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 2, pp. 982–993, 2021.

[32] J. Li, M. Song and L. Wang, et al, Hybrid artificial bee colony algorithm for a parallel batching distributed flow-shop problem with deteriorating jobs, *IEEE Transactions on Cybernetics*, vol. 50, no. 6, pp. 2425–2439, 2019.

[33] H. Cho, S. Bae and J. Kim, et al, Bi-objective scheduling for reentrant hybrid flow shop using Pareto genetic algorithm, *Computers & Industrial Engineering*, vol. 61, no. 3, pp. 529–541, 2011.

[34] K. Deb, A. Pratap and S. Agarwal, et al, A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[35] Y. Yuan, and H. Xu, Multiobjective flexible job shop scheduling using memetic algorithms, *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 1, pp. 336–353, 2013.

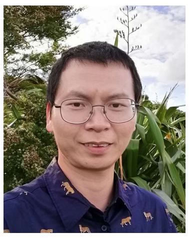

Pan Li received the MSc degree from Hebei University, Baoding, China, in 2022 and is currently pursuing the PhD degree at School of Cyber Science and Engineering, Southeast University, Nanjing, China. His research interests include computer vision and multivariate time series analysis.

Kai Di received the PhD degree from School of Computer Science and Engineering, Southeast University, Nanjing, China. He is currently a postdoctoral researcher of School of Computer Science and Engineering, Southeast University. He has authored several scientific articles in refereed journals and conference proceedings, such as *ACM Transactions on Autonomous and Adaptive Systems, and ACM Transactions on Intelligent Systems and Technology*. His current research focuses on multi-agent systems.

Xinlei Bai received the MSc degree from School of Cyber Science and Engineering, Southeast University, Nanjing, China. He research interests include multiplex networks and task allocation.

Fulin Chen received the MSc degree in automation control from Nanjing University of Posts and Telecommunications, Nanjing, China, in 2021. He is currently working toward the PhD degree at School of Cyber Science and Engineering, Southeast University, Nanjing, China. His research interests include data mining, multi-agent, and load forecasting.

Yuanshuang Jiang received the MSc degree from Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China, in 2021. He is currently working toward the PhD degree in computer science from Southeast University, Nanjing, China. His research interests include reinforcement learning and multi-agent systems.


Xiping Fu is a senior data scientist at PredictHQ in New Zealand. He earned the BS degree in mathematics from Zhejiang Normal University in 2007, the MSc degree in mathematics from Southeast University, China in 2010, and the PhD from University of Otago, New Zealand in 2016. His research interests encompass data mining, pattern recognition, and anomaly detection.