Optimal crane scheduling for tankhouse systems

Yuming Sun[a], Bianca Springub[b], Sofiya Onyshkevych[b], Christos Galanopoulos[b], Leonardo Salsano de Assis[b], John Wassick[a], Ignacio Grossmann[a] **(sequence is still being decided in process)**

[a] Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA

[b]Aurubis AG, Hovestrasse 50 20539 Hamburg, Germany

Highlights:

- Comparison of discrete-time mixed-integer linear programming (MILP), continuous-time MILP, and constraint programming (CP) models for single- and multi- crane scheduling in tankhouse systems.
- Introduction of novel machine-based decomposition strategy for constraint programming (CP) models.
- Real-world case studies demonstrate the effectiveness of the proposed solution strategy.
- The quantitative results confirm the performance of the proposed algorithm on industrial-scale problems.

Abstract

With the development of industry and automation, optimizing crane operations becomes increasingly important for improving productivity and reducing costs in manufacturing systems. In this paper, we address a crane scheduling problem in tankhouse systems, where operations are tightly integrated with the production process and the resource sharing is frequent. We propose discrete-time mixed-integer linear programming (MILP), continuous-time MILP, and constraint programming (CP) models to tackle this problem. It is found that the CP model has the best performance considering both single- and multi- crane conditions. To efficiently solve real-world problems, we propose a machine-based decomposition strategy with the CP framework, which solves 128 tasks in 34 seconds over 24 iterations. The results demonstrate the effectiveness of the proposed approach in handling large-scale, complex scheduling problems with high computational efficiency.

Keywords

Crane scheduling; Tankhouse; Mixed-integer linear programming; Constraint programming; Decomposition

1. Introduction

Cranes play an important role in lifting, delivery and transfer of goods in industrial and construction operations. With the development of industry and automation, cranes are widely used in construction, ports, and warehouse management. An efficient crane scheduling system can effectively save labor, electricity, time, and costs, as well as improve the quality of the product. In construction, where vertical lifting is a primary focus, effective crane scheduling optimizes automation, thereby enhancing both safety and operational efficiency (Zhu et al., 2023). For quay crane operations, which involve container transfers between vessels and land terminals, optimal scheduling improves overall performance, while balancing operational efficiency and energy consumption (Tan et al., 2021). In yard operations, where cranes handle container movements between storage blocks and input/output (I/O) points, realistic scheduling reduces weighted costs involving container tardiness, I/O point congestion, and crane travel time (Wang et al., 2025). In warehouse management, where cranes facilitate internal inventory movement, strategic scheduling streamlines logistic processes and reduces inventory holding costs (Peng et al., 2021). In this paper, our focus is on optimal crane scheduling in a manufacturing environment, with particular emphasis on tankhouse systems.

In manufacturing, cranes are responsible for handling materials across various stages of the production process. Different from common quay and yard cranes, crane operations in manufacturing feature tighter integration with production processes, stricter timing constraints, and more frequent resource sharing. One typical example is the steel-making process, where crane operations serve as a critical logistical component in the pre-steelmaking stage. Crane efficiency directly impacts converter utilization, production output, and overall operational costs (Xie et al., 2023). Similarly, cranes are also widely used for copper refining processes in electrolyte purification plants. The process, shown in Fig.1, involves casting anodes, transporting them to electrolysis cells for purification, and then returning used anodes to the converting stage for cooling and reuse, completing a closed loop material flow (Siitari, 2022).

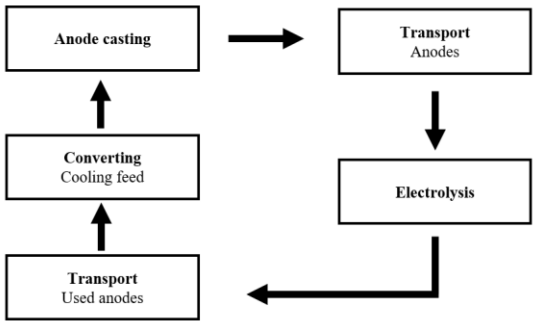

Fig.1. Copper refining processes.

Fig.2 shows the typical crane layout in a tankhouse, where the cranes hang and move along the track. The cranes must complete tasks at different locations, while ensuring non-intersecting track paths. For example, in Fig.2, track 2 should always be on the north of track 1. If crane 1 is assigned

a task further north, then tracks 2 - 4 should shift northward to leave space for track 1. The whole scheduling process is a dynamic process, requiring continuous adjustments to crane positions and task assignments to prevent conflicts, and ensure smooth operation across multiple tracks.

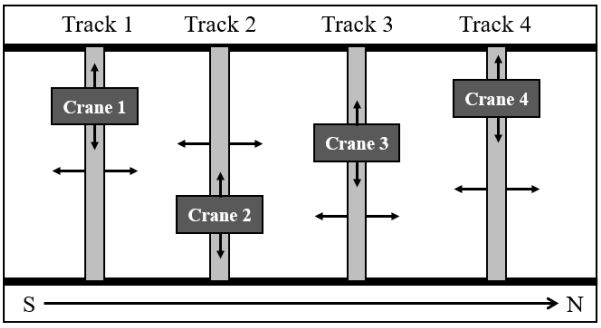

Fig.2. Crane movement in factories.

Such a crane scheduling problem is complex since it requires calculating a space-time trajectory for each crane, while also considering the order and timing between cranes (Aron et al., 2010). To solve the problem, the most common solution techniques include mathematical optimization models, simulation, evolutionary heuristics, and heuristics (Naeem et al., 2023). Each approach offers advantages and trade-offs in terms of scalability, accuracy, and data requirements. One approach to tackle such problems is to develop a tool relying on visualization of the process and graphically simulate the behaviors of the systems. This approach is effective in helping to identify unexpected problems and potential risks before construction (Kang Shih-Chung et al., 2009).

Another common approach is to use deep learning techniques. In this case, the advantage is that one can train the model with large-scale crane transportation data, which is more objective and sometimes more reliable than human experience (Feng et al., 2022). However, the difficulty for this method also lies in gaining such largescale data, since in practice there is often a lack of historical data of good quality.

Heuristic methods in crane scheduling problems improve efficiency to find solutions (Bierwirth and Meisel, 2009). They work well with models of large scale. However, compared to mathematical optimization methods, they are more likely to give suboptimal solutions.

Mixed-integer linear programing (MILP) models are useful in scheduling problems since they handles complex constraints with precision to provide optimal solutions (Meng et al., 2023). Several techniques have been proposed to solve MILP models more efficiently, including branch-and-cut algorithm (Moccia et al., 2006), Benders decomposition (Emde, 2017), and dynamic programming (Peterson et al., 2014). Furthermore, this approach does not rely heavily on historical data, and therefore is suitable for a wide range of real-life applications.

The structure of this paper is organized as follows. Section 1 provides an introduction and background on crane scheduling problems. Section 2 presents a literature review on related work and includes description of similar problems and common methodology. Section 3 defines the

problem and offers a general problem statement. Section 4 gives a motivating example of the tankhouse layout and operations in copper purification process. Section 5 describes formulations for discrete-time MILP, continuous-time MILP, and constraint programming (CP) models for the given problems. Section 6 proposes a decomposition strategy as a more efficient approach to solve the problem. Section 7 shows the results of the proposed models on small, medium, and real-world examples. Section 8 summarizes our work in conclusions, and discusses future work.

2. Literature review

Crane scheduling has long been a critical problem across industries, focusing on problems such as quay crane (QC) scheduling, yard crane (YC) scheduling, and yard truck (YT) dispatching (H. -P. Hsu et al., 2022). Common objective functions include minimizing the number of container moves and internal vehicle travel distances, the task completion time, the due date satisfaction, and yard space utilization. Factors such as equipment utilization, environmental concerns, and cost have received relatively less attention (Kizilay and Eliiyi, 2021).

Early studies have focused on port operations and formulated models under simplified conditions. Daganzo (1989) proposes a mixed-integer linear programming (MILP) approach to crane scheduling at ports, introducing a simple static model as a foundation for dynamic cases with berth limitations, developing scheduling principles that minimize ship delay and crane idle time. Building on this foundation, and due to the operational nature of crane scheduling problems, discrete-time models have been developed. Li et al. (2009) introduce discrete-time formulations for yard crane scheduling, where heuristics and a rolling-horizon algorithm significantly reduce model complexity and computational time. Later, Agra and Oliveira (2018) have advanced discrete modeling further by proposing a time–space discretized reformulation for the integrated berth and quay crane assignment problem, avoiding big-M constraints and facilitating exact solution methods through a branch-and-cut algorithm supported by rolling-horizon heuristics. As research has progressed, continuous-time models have been introduced to make models more efficient. Li et al. (2015) develop a continuous-time model combined with a rolling-horizon algorithm, significantly reducing model size and reducing computational time from days to seconds, with a higher solution quality than existing heuristics. However, the model relies on binary variables to determine time overlap and crane conflicts, which simplifies collision detection of the cranes, but fails to capture dynamics. To better handle such spatial and temporal interactions, constraint programming (CP) provides a more expressive framework that supports flexible modeling of sequencing, resource sharing, and mutual exclusion. Qin et al. (2020) propose a hybrid MIP/CP solution strategy that effectively solves large-scale instances involving up to 1,000 containers, 6 quay cranes, 36 yard trucks, and 15 yard cranes in 2 minutes with an optimality gap of less than 3.31%. This strategy takes advantage of the strengths of the branch-and-cut (B&C) algorithm for MIP and the branch-and-price (B&P) algorithm for CP. However, the model makes unidirectional assumptions, which may be suitable for port operations with linear container flow, but not for manufacturing environments where cranes often need to shuttle back and forth between stations.

The unidirectional limitation is then solved by a new CP model despite the exponential increase in search space (Unsal and Oguz, 2013).

While most existing studies focus on crane scheduling in port and yard operations, we shift our attention to manufacturing systems, where crane coordination introduces different scheduling requirements due to the complexity of material flows, station layouts, and dynamic task requirements. In response to these distinct requirements, several studies have explored crane scheduling in manufacturing environments. Tanizaki et al. (2006) present a model with crane interference and applied a heuristic method on an enumeration tree for crane assignment, based on depth-first and width-first search. The method first determines task sequences heuristically based on the latest due time before assigning cranes, which limits its applicability in scenarios where task precedence is not clearly defined. Liu et al. (2023) develop a low-carbon scheduling model for flexible manufacturing and crane transportation, and demonstrate that the proposed differential evolution (DE) with firefly algorithm (FA) and a collaborative state optimization strategy (CSOS) reduces machining and transportation energy consumption by 25.17% and 34.52%, respectively. The limitation of this study is that it considers only a single crane for operations, without accounting for the interactions and potential conflicts among multiple cranes in a shared workspace. By narrowing the focus to tankhouse systems, few studies have addressed crane scheduling, with most research focusing on process operations (Fuke et al., 2025) or electrochemistry (S. Grobbelaar et al., 2023).

In this paper, we focus on crane scheduling problems for tankhouse systems. We first compare the performance of discrete-time MILP, continuous-time MILP, and constraint programming (CP) models for single and multi- crane scheduling problems. We then apply a machine-based decomposition strategy with CP models for large-scale problems. Our findings complement existing research by providing a comprehensive comparison of modeling approaches under tankhouse-specific constraints. Specifically, we demonstrate the effectiveness of decomposition in enhancing the scalability of constraint programming (CP) models for large-scale, interference-constrained, and non-unidirectional scheduling.

3. Problem statement and motivation

3.1. The copper purification process in a tankhouse

The primary goal of the smelting process for copper production is to produce high-purity copper anodes. The anodes are mainly composed of copper. However, there are small amounts of impurities. Therefore, these anodes need to go through an electrolysis process at the tankhouse to refine copper and recover a solution with all other elements that are not copper. During the electrolysis process, copper ions are diluted and attached to a steel sheet that is used as the cathode (Fig.3). This copper purification process is conducted by cranes that operate on individual cells.

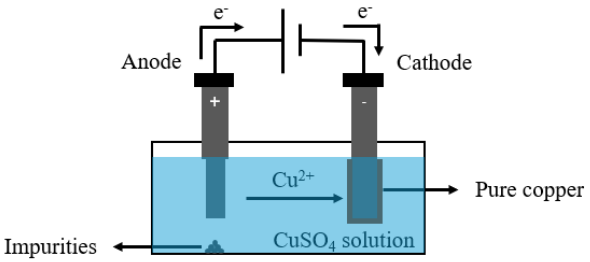

Fig.3. Schematic diagram of a cell.

3.2. The tankhouse crane scheduling problem

In the crane scheduling problem, the following is assumed to be given: 1) the tankhouse layout divided into several blocks, 2) several cranes that are responsible for pickup, transfer, and delivery of objects, 3) a set of tasks that need to be completed, and 4) several machines that process objects. Additionally, there are some unique requirements on the operations for the copper purification process:

1) Cells provide the locations for the placement of copper anodes and metal sheets. During the electrolysis process, there are several crops for the same solution. Therefore, for each cell, there is a sequence of tasks in an operation cycle, for example, cathode operations and anode operations.
2) Machines are shared across the block and cannot exceed their maximum processing capacity.
3) Cranes can operate on either the same cell or on different cells. However, this should not happen at the same time. Although there is a certain sequence of tasks in one cell, tasks in different cells have no sequencing limitations. For example, after finishing cell one task one, the crane needs to decide whether the next step should be cell one task two or cell two task one.
4) For each block, one single crane or multiple cranes could be available. Cranes are non-unidirectional and move between cells and machines, transferring cathodes and anodes. Cranes in the same block share a common track, and require a safety distance to avoid interference. Cranes across different blocks work independently.

The goal of the scheduling problem is to allocate cranes efficiently to maximize productivity. This means ensuring the machines are working at their full capacities and do not have to wait. Furthermore, the cell is inactive if the crane is operating on it. Since cells in one block share the same electric circuit, this means that the electricity in the whole block should be shut down, which leads to stopping the electrolysis process for multiple cells. This might result in a worse result. Therefore, the objective is to minimize the time in which cells are inactive, which is equivalent to the completion time for all cells.

4. Motivating examples

4.1. Tankhouse layout

To illustrate the tankhouse system, we provide a small illustrative example (Fig.4). The system contains 4 blocks, some with single-crane and others with multi-crane configurations. All cranes transport anodes and steel sheets between cells and a shared machine. Given that the machine has only one slot, uses across blocks can improve efficiency. For example, while the cell in block 1 is using the machine, cranes in blocks 2 – 4 can perform other subtasks that do not require the machine. Our goal is to optimize the crane movement and find the sequence of tasks so that the machines are used to the fullest extent and the operation time is minimized.

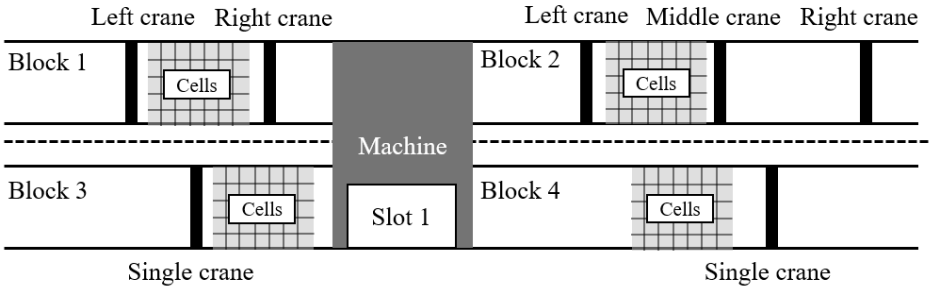

Fig.4. An illustrative example of the tankhouse layout.

4.2. Operations in copper purification process

Before presenting the modeling of crane scheduling problems, we provide an illustrative description of the detailed operations in copper purification process. Each cell uses a steel sheet and a copper. As the electrolysis process begins, copper ions from the anode dissolve into the electrolyte and are deposited onto the steel sheet, gradually forming the cathode-high-purity copper on the steel sheet. There are mainly two kinds of operations during this process, the cathode operations (Fig.5) and the anode operations (Fig.6). These two kinds of operations are both conducted by cranes.

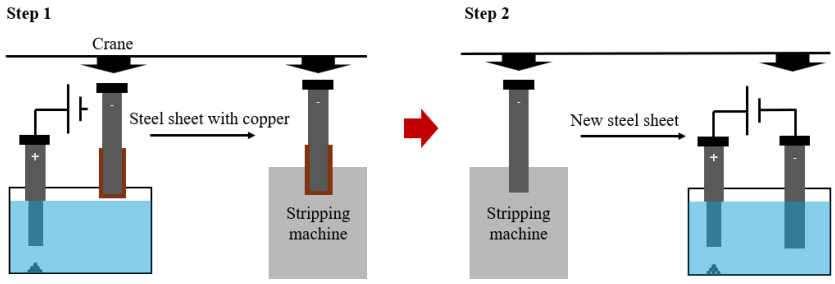

Fig.5. The cathode operations for one cell.

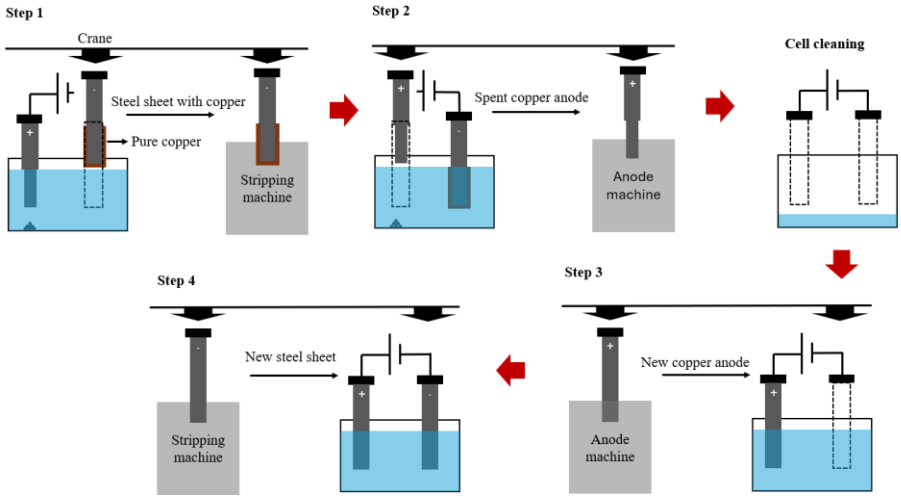

Fig.6. The anode operations for one cell

The cathode operations only involve cathode exchange, which consists of two steps. The first step is to take the steel sheet with pure copper from the cell to the stripping machine. The second step is to bring the new steel sheet from the stripping machine back to the cell.

The anode operations are more complex, involving four steps in the following way. The electricity in the whole group with the operated cell is shut down. First, the steel sheet with cathode is transported to the stripping machine, and the spent copper anode is transported to the anode machine. Then after both electrodes are removed, the cell is cleaned before new electrodes are inserted. Finally, the new copper anode and the new steel sheet are transported to the cell from the anode machine and the stripping machine correspondingly. The electricity in the section is turned on again only when the operations for every cell in the group are finished. Furthermore, due to the specific structure of the cell, the cathode physically obstructs the anode. Therefore, the cathode must be removed before the anode, and the anode must be inserted before the cathode.

For each cell, cathode operations and anode operations are carried out at regular intervals, with at most one type of operation performed per day. To better illustrate the process, Fig.7 depicts the crane movements during operations. The cathode tasks involve steps 1 and 4, while anode tasks include all four steps.

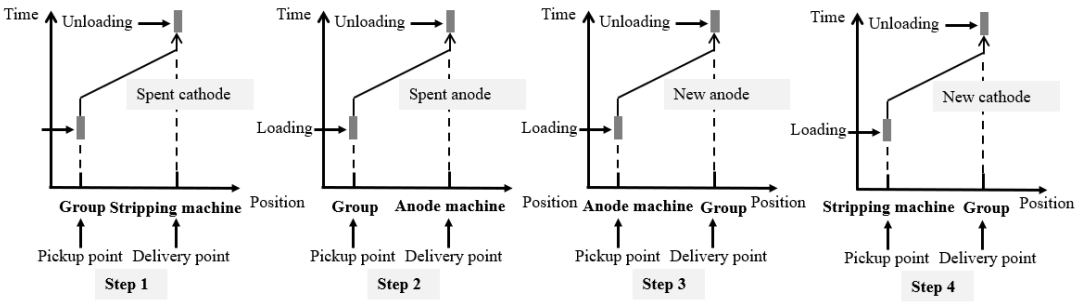

Fig.7. Crane movements for cathode and anode operations.

5. Model formulation

5.1. Outline of modeling approaches

In this section, we introduce the mathematical formulations of the crane scheduling problem. We explore different ways of modeling the problem, and indicate advantages and limitations in them.

We first present a full-size discrete-time mixed-integer linear programming (MILP) model with all the necessary constraints for cranes, tasks, and machines. We then reformulate the problem as the continuous-time MILP model to reduce the complexity of the model. However, in this case, we lose the ability to model the cranes with multiple cranes due to the failure in avoiding crane interferences. To overcome this, we combine both models into a hybrid model. Due to the characteristics of the hybrid model, particularly its large number of discrete variables, which lead to a weak LP relaxation, we finally propose a constraint programming approach to model the underlying process. This approach is well-suited for handling integer variables and complex logical constraints.

We make the following assumptions according to the features of tankhouse operations for the copper purification process.

1) Operations in each block start at the same time.
2) All cells have the same properties and sequence of tasks. In each cell, the next task can only start after the previous task is completed.
3) The pickup and delivery positions for tasks in cells are known.
4) Cells in the same block share the same electrical circuit. Therefore, the operations on one cell will result in the electrical shutdown of the entire block, which means all cells in this block will be inactive that would affect the electrolysis process for the cells. Under this condition, minimizing the sum of completion times for all blocks becomes a meaningful objective.
5) All cranes are the same, which means that they have the same velocity, pickup time, and delivery time. The velocity equals the smallest time step in the model.
6) The time that cranes move vertically along the track can be ignored compared to the time for cranes to move horizontally between cells and machines.
7) Each machine can only deal with one task at a time.
8) If the machine is in use, the crane should wait above until the machine becomes available.
9) The machine processes the object when the crane delivers it without any waiting time.
10) There is no human factor, meaning that all cleaning times are constant.

We define the following parameters and sets,

T: The set of discrete time points in the time horizon, where the time horizon represents the upper bound on the total time required to complete all tasks.

t, t', t'': The index of discrete time, $t, t', t'' \in T$.

N: The set of all independent blocks in the tankhouse.

n: The index of blocks, $n \in N$.

C^n: The set of cranes in block n.

c, c': The index of cranes, $c, c' \in C^n$.

J^n: The set of cells in block n.

j, j': The index of cells, $j, j' \in J^n$.

K: The set of all steps for one cell.

k, k': The index of steps for one cell, $k, k' \in K$.

I^n: The set of possible positions for cranes in block n.

i: The index of crane positions.

$Posp_{n,k}$: Pickup position of step k for cells in block n.

$Posd_{n,k}$: Delivery position of step k for cells in block n.

$PosI_c$: Initial position for crane c.

L_{safety}: Safety distance between cranes.

Q: The set of all shared machines.

q: The index of shared machines, $q \in Q$.

k_q: A certain step in one cell that utilizes the machine q, $k_q \in K$.

QP_q: Processing time of machine q for one task.

P: Pickup duration.

D: Delivery duration.

M: A large value for big-M constraints.

Next we define the binary variables,

$y_{c,j,k,t}$: If crane c picks up cell j step k at time t.

$z_{c,j,k,t}$: If crane c delivers cell j step k at time t.

$x_{c,i,t}$: If crane c locates at position i at time t.

$w_{q,j,j'}$: If cell j uses the machine q before cell j'.

Finally, we define the automative decision variables,

TC: Total completion time for all blocks.

T^n: The completion time for all cells in block n.

5.2. Discrete-time MILP

In this model, we need to determine the following decision variables: 1) the movement paths and task assignments for cranes in each block, 2) the sequences of tasks between cells, 3) the allocation of the use of shared machines, and 4) the minimum completion for all cells. Fig.8 shows a schematic representation of the discrete-time MILP model, with detailed equations explained below.

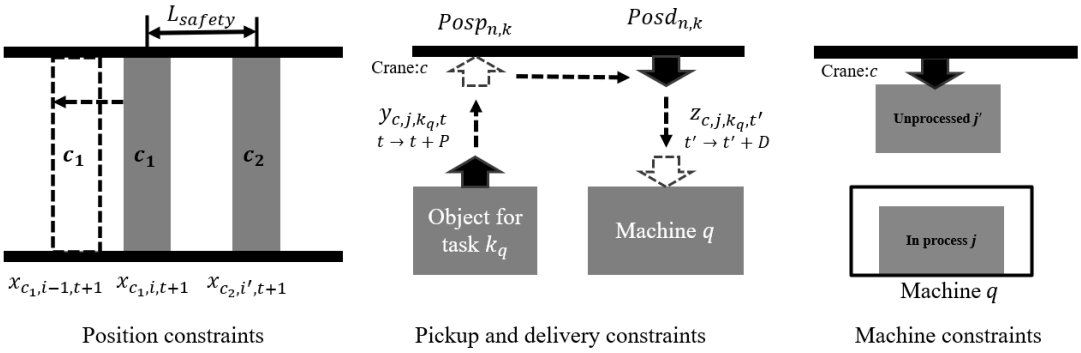

Fig.8. Discrete-time MILP model schematic diagrams.

Equation (1) corresponds to the objective function. Our goal is to minimize the total completion time TC, which equals the sum of completion time in each block as stated in constraint (2). The completion time for each block should be at least equal to the sum of the times needed to start the final delivery operation and the time required for the delivery itself as stated in constraint (3), which is activated if crane c delivers all tasks for cell j step k at time t (i.e. $z_{c,j,k,t} = 1$).

$$\min\ TC \tag{1}$$

$$s.t. TC \geq \sum_{n \in N} T^n \qquad \forall\, n \in N \tag{2}$$

$$T^n \geq (t + D) \cdot z_{c,j,k,t} \qquad \forall\, n \in N, \forall\, c \in C^n, \forall j \in J^n, \forall k \in K, \forall t \in T \tag{3}$$

Constraints (4) – (10) correspond to the constraints for the cranes. Constraint (4) defines the initial positions of the cranes. Constraint (5) ensures the crane to stay at one position at a time. Constraint (6) defines the maximum velocity of cranes. Constraints (7) and (8) ensure the cranes stay at the pickup or delivery location for long enough to finish the pickup or delivery operations. For each crane, it can only do one task at a time as stated in constraints (9) and (10).

$$x_{c,PosI_c,0} = 1 \qquad \forall\, c \in C^n \tag{4}$$

$$\sum_{i \in I^n} x_{c,i,t} = 1 \qquad \forall c \in C^n, \forall t \in T \tag{5}$$

$$x_{c,i,t} \leq x_{c,i-1,t+1} + x_{c,i,t+1} + x_{c,i+1,t+1} \qquad \forall c \in C^n, \forall i \in I^n, \forall t \in T \tag{6}$$

$$y_{c,j,k,t} \leq x_{c,Posp_{n,k},t'} \qquad \forall n \in N, \forall c \in C^n, \forall j \in J^n, \forall k \in K, \forall t, t' \in T, 0 \leq t' - t \leq P \tag{7}$$

$$z_{c,j,k,t} \leq x_{c,Posd_{n,k},t'} \qquad \forall n \in N, \forall c \in C^n, \forall j \in J^n, \forall k \in K, \forall t, t' \in T, 0 \leq t' - t \leq D \tag{8}$$

$$y_{c,j,k,t} + z_{c,j,k,t''} + y_{c,j',k',t'} \leq 2 \tag{9}$$

$$\forall c \in C^n, \forall j, j' \in J^n, \forall k, k' \in K, \forall t, t', t'' \in T, j \neq j' \text{ or } k \neq k', t \leq t' \leq t''$$

$$y_{c,j,k,t} + z_{c,j,k,t''} + y_{c,j',k',t'} \leq 2 \tag{10}$$

$$\forall c \in C^n, \forall j, j' \in J^n, \forall k, k' \in K, \forall t, t', t'' \in T, j \neq j' \text{ or } k \neq k', t \leq t' \leq t''$$

To be more specific, constraints (9) and (10) are derived from the logical forms (11) and (12)(Raman and Grossmann 1994), which means that for one crane, during the pickup and delivery status of a single task, it cannot conduct any other tasks.

$$y_{c,j,k,t} \wedge z_{c,j,k,t''} \Rightarrow \neg y_{c,j',k',t'} \tag{11}$$

$$\forall c \in C^n, \forall j, j' \in J^n, \forall k, k' \in K, \forall t, t', t'' \in T, j \neq j' \text{ or } k \neq k', t \leq t' \leq t''$$

$$y_{c,j,k,t} \wedge z_{c,j,k,t''} \Rightarrow \neg z_{c,j',k',t'} \tag{12}$$

$$\forall c \in C^n, \forall j, j' \in J^n, \forall k, k' \in K, \forall t, t', t'' \in T, j \neq j' \text{ or } k \neq k', t \leq t' \leq t''$$

Constraint (13) is an additional constraint for blocks with multiple cranes. It ensures the safety distance and relative positions between cranes, for example, the left crane is always on the left of the right crane. In this way, the cranes will not overlap during operation.

$$x_{c,i,t} + x_{c',i',t} \leq 1 \qquad \forall c, c' \in C^n, \forall i \in I^n, \forall t \in T, c > c', i' < i + L_{safety} \tag{13}$$

Constraints (14) – (18) are constraints for the tasks. For each object, it should be picked up or delivered once as enforced by constraints (14) and (15). The pickup and delivery of one object should be completed by the same crane as enforced by constraint (16). Constraint (17) ensures that the pickup of each object should be conducted before delivery. For tasks inside the same cell, the next step begins after the previous step is finished as enforced by constraint (18).

$$\sum_{c \in C^n, t \in T} y_{c,j,k,t} = 1 \qquad \forall j \in J^n, \forall k \in K \tag{14}$$

$$\sum_{c \in C^n, t \in T} z_{c,j,k,t} = 1 \qquad \forall j \in J^n, \forall k \in K \tag{15}$$

$$\sum_t y_{c,j,k,t} = \sum_t z_{c,j,k,t} \qquad \forall c \in C^n, \forall j \in J^n, \forall k \in K, \forall t, t' \in T, t \neq t' \tag{16}$$

$$y_{c,j,k,t'} + z_{c,j,k,t} \leq 1 \qquad \forall c \in C^n, \forall j \in J^n, \forall k \in K, \forall t, t' \in T, t \leq t' \tag{17}$$

$$\sum_{c \in C^n} y_{c,j,k',t'} + \sum_{c \in C^n} z_{c,j,k,t} \leq 1 \quad \forall i \in I^n, \forall k, \forall k' \in K, \forall t, t' \in T, t' \leq t, k < k' \tag{18}$$

Constraints (19) – (20), which are big-M constraints, are the constraints on the use of the machines. Since the machine can only deal with one object at a time, the time gap between the use of machines should be longer than the processing time of machines. If the machine is in use, the cranes cannot unload the object, and must wait overhead of the machine. Therefore, there should be a sequential order for the unloading of objects.

$$(t + D) \times \sum_{c \in C^n} z_{c,j,k_q,t} + QP_q \leq (t' + D) \times \sum_{c \in C^n} z_{c,j',k_q,t'} + M \cdot (1 - w_{q,j,j'}) \tag{19}$$
$$\forall c \in C^n, \forall j, j' \in J^n, \forall t, t' \in T, \forall q \in Q, j \neq j'$$

$$(t' + D) \times \sum_{c \in C^n} z_{c,j',k_q,t'} + QP_q \leq (t + D) \times \sum_{c \in C^n} z_{c,j,k_q,t} + M \cdot w_{q,j,j'} \tag{20}$$
$$\forall c \in C^n, \forall j, j' \in J^n, \forall t, t' \in T, \forall q \in Q, j \neq j'$$

Constraint (21) specifies the domains of the decision variables.

$$y_{c,j,k,t}, z_{c,j,k,t}, x_{c,i,t}, w_{q,j,j'} \in \{0,1\}, TC, T^n \geq 0 \tag{21}$$

In summary, the discrete-time MILP model is given by the objective function in equation (1), subject to the constraints (2) - (10) and (13) – (21).

5.3. Continuous-time MILP

The discrete-time MILP model can be computationally expensive since the extension of the time horizon leads to an exponential increase in model complexity due to the growing number of discrete time variables.

In order to reduce the size of the model, we transform the discrete-time MILP model into the continuous-time MILP model. We convert the discrete time points into continuous time, and transform the position constraints into time constraints using velocity. Instead of focusing on what the cranes do at each time point, we place the emphasis on the start time and end time of each task for each crane. Another advantage is that not being limited to the discrete time points, we have a wider range of velocity values. As will be shown later in the computational results, this model works well for blocks with a single crane. The constraints for the continuous-time MILP model are presented below, along with an illustrative diagram in Fig.9, followed by some additional parameters and variables.

v: Crane velocity.

$r_{j,j',k,k'}$: If cell j step k starts before cell j' step k'.

$sp_{j,k}$: Start time for the pickup of cell j step k.

$sd_{j,k}$: Start time for the delivery of cell j step k.

$ep_{j,k}$: End time for the pickup of cell j step k.

$ed_{j,k}$: End time for the delivery of cell j step k.

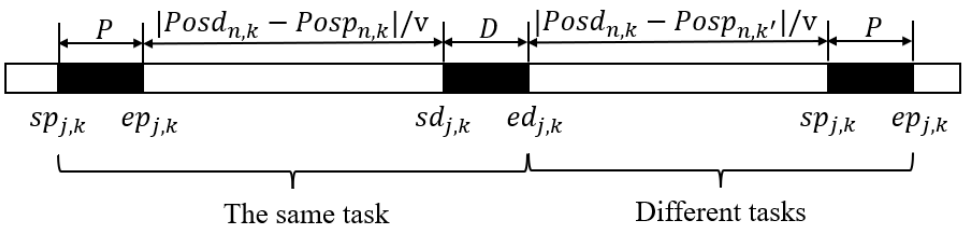

Fig.9. Continuous-time MILP model schematic diagram.

After converting the location information into time information, some of the constraints in the discrete-time MILP model can be replaced by continuous-time constraints. Constraint (3) can be replaced by constraint (22), with continuous variables for the completion time in blocks with one crane. Furthermore, constraints (4) – (21) can be replaced by constraints (23) – (32) for crane movement and task assignments.

$$T^n \geq ed_{j,k} \qquad \forall n \in N, \forall j \in J^n, \forall k \in K \tag{22}$$

Constraints (23) - (24) specify the start and end time for pickup and delivery.

$$ep_{j,k} \geq sp_{j,k} + P \qquad \forall j \in J^n, \forall k \in K \tag{23}$$

$$ed_{j,k} \geq sd_{j,k} + D \qquad \forall j \in J^n, \forall k \in K \tag{24}$$

Considering the initial conditions, constraint (25) specifies the earliest start time for each task, where $PosI_c$ and $Posp_{n,k}$ are given position parameters.

$$sd_{j,k} \geq \left|PosI_c - Posp_{n,k}\right|/v \qquad \forall n \in N, \forall c \in C^n, j \in J^n, \forall k \in K \tag{25}$$

Constraints (26) – (27) define the task sequences for one cell, accounting for the transfer time in one task and between tasks.

$$sd_{j,k} \geq ep_{j,k} + \left|Posd_{n,k} - Posp_{n,k}\right|/v \qquad \forall n \in N, \forall j \in J^n, \forall k \in K \tag{26}$$

$$sp_{j,k'} \geq ed_{j,k} + \left|Posd_{n,k} - Posp_{n,k'}\right|/v \qquad \forall n \in N, \forall j \in J^n, \forall k, k' \in K, k < k' \tag{27}$$

Constraints (28) – (29) are the big-M constraints for tasks in different cells, so that tasks between different cells can be crossed.

$$ed_{j,k} + \left|Posd_{n,k} - Posp_{n,k'}\right|/v \leq sp_{j',k'} + M \cdot \left(1 - r_{j,j',k,k'}\right) \quad (28)$$

$$\forall n \in N, \forall j, j' \in J^n, \forall k, k' \in K, j \neq j'$$

$$ed_{j',k'} + \left|Posd_{n,k'} - Posp_{n,k}\right|/v \leq sp_{j,k} + M \cdot r_{j,j',k,k'} \quad (29)$$

$$\forall n \in N, \forall j, j' \in J^n, \forall k, k' \in K, j \neq j'$$

Constraints (30) – (31) are the constraints for the use of machines.

$$ed_{j,k_q} + QP_q \leq ed_{j',k_q} + M \cdot (1 - w_{q,j,j'}) \qquad \forall j, j' \in J^n, \forall q \in Q, j \neq j' \quad (30)$$

$$ed_{j',k_q} + QP_q \leq ed_{j,k_q} + M \cdot w_{q,j,j'} \qquad \forall j, j' \in J^n, \forall q \in Q, j \neq j' \quad (31)$$

Constraint (32) specifies the domains for the additional decision variables.

$$sp_{j,k}, sd_{j,k}, ep_{j,k}, ed_{j,k} \geq 0, r_{j,j',k,k'} \in \{0,1\} \quad (32)$$

In summary, the continuous-time MILP model is given by the objective function in equation (1), subject to the constraints (2) and (22) – (32).

5.4. Constraint programming (CP) model

5.4.1 Introduction of the CP model

Constraint programming (CP) is a logic-based method that utilizes implicit constraints and logical inference for optimization (Hooker, 2002). Constraint Programming does not follow a fixed canonical form, allowing it to flexibly accommodate a wide range of variable types and constraint structures. CP is often used for constraint satisfaction problems (CSP) and constraint optimization problems (COP), defined by a collection of variables, each associated with a domain and constraints that restrict combinations of their values (Rossi et al., 2008). While CP is primarily designed to find feasible solutions that satisfy all given constraints, it can also be extended to handle optimization by searching through a sequence of feasible solutions that progressively improve the objective function. A key feature of CP is that it operates directly on discrete variables, rather than relying on continuous relaxations (Pesant, 2014). Its search method is primarily driven by domain reduction, constraint propagation, and tree search to efficiently explore the solution space (Hooker, 2002). These characteristics make CP suitable for crane scheduling problems, where tasks, resources, and time slots are naturally discrete, and the problem involves complex constraints such as task sequencing, non-overlapping crane operations, and shared use of machines.

5.4.2 Formulation of the CP model

To implement CP models effectively, we apply the Optimization Programming Language (OPL), which provides a high-level modeling framework tailored for constraint-based problem solving (Van Hentenryck et al., 1999). OPL supports key CP functionalities such as decision variable

declarations, constraint definitions, and search strategies, allowing complex scheduling logic to be expressed in a compact and intuitive manner. In OPL CP, there is a special type of decision variable called interval variables. These interval variables are both variables and constraints. A typical interval variable can be written in the form as $NewIntervalVar(start, duration, end, name)$, where $end - start == duration$. It should also be mentioned that $start, duration, end$ can be constraints or variables. In particular, when certain tasks are optional, they are modeled as optional interval variables using the function as $NewOptionalIntervalVar(start, duration, end, is\_present, name)$, where $is\_present$ indicates if the interval is active or not. The interval variables are widely used in scheduling problems, often together with constraints related to sequences, such as $NoOverlap$, $EndBeforeStart$, and $EndAtStart$. The followings are the OPL functions needed in our crane scheduling problem, together with illustrative diagrams in Fig.10 and some additional variables.

OPL functions

$ExactlyOne(x_1, x_2 \dots x_n)$: Only one variable is $True$ among a set of variables x.

$StartOf(z)$: The start time for interval variable z.

$EndOf(z)$: The end time for interval variable z.

$LengthOf(z)$: The length of interval variable z, i.e., the time gap between its start and end time.

$Presenceof(z)$: The presence status of interval variable z.

$NoOverlap(\{z_1, z_2 \dots z_n\}, t_{i,j})$: There should be no overlaps for interval variables z, with the minimum time gap of $t_{i,j}$ between z_i and z_j.

$EndBeforeStart(x, y, t)$: y can start t time after the end of x, that is $EndOf(x) + t \leq StartOf(y)$.

$(X)OnlyEnforceIf(x_1, x_2, \dots, x_n)$: If all Boolean variables $x_1, x_2, \dots, x_n$ are True, then constraint X is active.

$AbsEquality(X)$: The absolute value of the expression of X.

Additional variables

$Pos_{c,t}$: The position for crane c at time t.

$Intv_{j,k}^{c}$: Interval variables during which crane c transfers the object in cell j step k.

$Intv_{j,k}$: Interval variables during which cell j step k is conducted.

$ass_{c,j,k}$: If cell j step k is assigned to crane c.

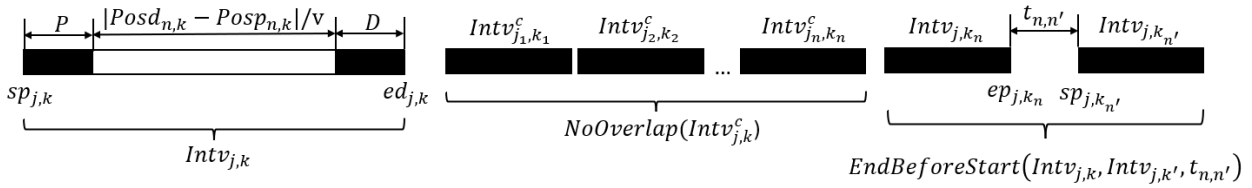

Fig.10. CP model schematic diagrams.

Constraint (32) represents the objective function for the CP model, which is the same as constraints (1) in the discrete-time MILP model. Constraints (33) – (34) define the completion time in total and the separate completion time in each block.

$$\min\ TC \tag{32}$$

$$s.t. TC \geq \sum_{n \in N} T^n \qquad \forall\, n \in N \tag{33}$$

$$T^n \geq\ ed_{j,k} \qquad \forall\, n \in N, \forall j \in J^n, \forall k \in K \tag{34}$$

Constraint (35) defines the initial position for the cranes.

$$Pos_{c,0} = PosI_c \qquad \forall c \in C^n \tag{35}$$

Constraint (36) states the relationship between the integer position variables $Pos_{c,t}$ and the binary position variable $x_{c,i,t}$.

$$Pos_{c,t} = i \cdot \sum_{i \in I} x_{c,i,t} \qquad \forall c \in C^n, \forall i \in I^n, \forall t \in T \tag{36}$$

Constraint (37) is identical to constraint (5) to ensure the crane only stays at one position at a time.

$$ExactlyOne\big(\{x_{c,i,t} | i \in I\}\big) \qquad \forall c \in C^n, \forall t \in T \tag{37}$$

Constraint (38) defines non-interference constraints for multiple cranes.

$$Pos_{c,t} + L_{safety} \leq Pos_{c',t} \qquad \forall c, c' \in C^n, \forall i \in I^n, \forall t \in T, c < c' \tag{38}$$

Constraint (39) ensures the operation time for different tasks assigned to the same crane do not overlap.

$$\{(sp_{j,k} \geq ed_{j',k'}) \vee (sp_{j',k'} \geq ed_{j,k})\} OnlyEnforceIf (ass_{c,j,k} = ass_{c,j',k'}) \tag{39}$$

$$\forall c \in C^n, \forall j, j' \in J^n, \forall k, k' \in K, j \neq j'\ or\ k \neq k'$$

Similarly, constraints (40) – (42) specify the start time, end time, time length, and the activation conditions of the interval variable $Intv_{j,k}$.

$$StartOf\big(Intv_{j,k}\big) = sp_{j,k} \qquad \forall j \in J^n, \forall k \in K \tag{40}$$

$$EndOf\left(Intv_{j,k}\right) = ed_{j,k} \qquad \forall j \in J^n, \forall k \in K \tag{41}$$

$$LengthOf\left(Intv_{j,k}\right) = P + D + \left|Posd_{n,k} - Posp_{n,k}\right|/v \qquad \forall n \in N, \forall j \in J^n, \forall k \in K \tag{42}$$

Constraint (43) defines the sequence for steps in the same cell.

$$EndBeforeStart\left(Intv_{j,k}, Intv_{j,k'}, 0\right) \qquad \forall j \in J^n, \forall k, k' \in K, k < k' \tag{43}$$

Constraints (44) – (45) use the logic form to ensure that the cranes stay at right positions during pickup and delivery. Specifically, if the pickup and delivery duration of operations can be ignored, we can apply constraints (46) – (47) to reduce the model complexity.

$$\left(\{Pos_{c,t} \middle| sp_{j,k} \le t \le ep_{j,k}\} = Posp_{n,k}\right) OnlyEnforceIf\left(ass_{c,j,k} = 1, sp_{j,k} = t\right) \tag{44}$$
$$\forall n \in N, \forall c \in C^n, \forall j \in J^n, \forall k \in K, \forall t \in T$$

$$\left(\{Pos_{c,t} \middle| sd_{j,k} \le t \le ed_{j,k}\} = Posd_{n,k}\right) OnlyEnforceIf\left(ass_{c,j,k} = 1, sp_{j,k} = t\right) \tag{45}$$
$$\forall n \in N, \forall c \in C^n, \forall j \in J^n, \forall k \in K, \forall t \in T$$

$$\left(Pos_{c,t} = Posp_{n,k}\right) OnlyEnforceIf\left(ass_{c,j,k} = 1, sp_{j,k} = t\right) \tag{46}$$
$$\forall n \in N, \forall c \in C^n, \forall j \in J^n, \forall k \in K, \forall t \in T$$

$$\left(Pos_{c,t} = Posd_{n,k}\right) OnlyEnforceIf\left(ass_{c,j,k} = 1, sd_{j,k} = t\right) \tag{47}$$
$$\forall n \in N, \forall c \in C^n, \forall j \in J^n, \forall k \in K, \forall t \in T$$

Constraint (48) defines the velocity of the cranes, and constraint (49) makes sure that the time gap between deliveries of objects using the same machine is larger than the processing time of the machine.

$$AbsEquality\left(Pos_{c,t} - Pos_{c,t+1}\right) \le v \qquad \forall c \in C^n, \forall t \in T \tag{48}$$

$$AbsEquality\left(ed_{j,k_q} - ed_{j',k_q}\right) \ge QP_q \qquad \forall j, j' \in J^n, q \in Q, k_q \in K, j \ne j' \tag{49}$$

Constraint (50) defines the domains for the additional decision variables.

$$Pos_{c,t} \ge 0, ass_{c,j,k} \in \{0,1\} \tag{50}$$

We should note that, for single-crane scheduling problems, constraint (39) can be replaced by constraints (51) – (55) to simplify the model. Constraints (51) – (54) are constraints to define the start time, end time, time length, and presence of the interval variable $Intv^c_{j,k}$ separately. $Intv^c_{j,k}$ can only be applied to single-crane conditions since the time length for the transfer of each task is a fixed value, without possible waiting time for yielding or to avoid interference.

$$StartOf\left(Intv_{j,k}^{c}\right) = sp_{j,k} \qquad \forall c \in C^n, \forall j \in J^n, \forall k \in K \tag{51}$$

$$EndOf\left(Intv_{j,k}^{c}\right) = ed_{j,k} \qquad \forall c \in C^n, \forall j \in J^n, \forall k \in K \tag{52}$$

$$LengthOf\left(Intv_{j,k}^{c}\right) = P + D + \frac{\left|Posd_{n,k} - Posp_{n,k}\right|}{v} \qquad \forall n \in N, \forall c \in C^n, \forall k \in K \tag{53}$$

$$Presenceof\left(Intv_{j,k}^{c}\right) = 0 \qquad \forall c \in C^n, \forall j \in J^n, \forall k \in K \tag{54}$$

Constraint (52) ensures that the tasks assigned to the same crane will not overlap with each other.

$$NoOverlap\left(Intv_{j,k}^{c}\right) \qquad \forall c \in C^n \tag{55}$$

In summary, the CP model is defined by the objective function (32), subject to the constraints (33) – (55).

6. Solution strategy

Solving the full-time industrial tankhouse crane scheduling problem is computationally expensive, whether using the discrete-time MILP, the continuous-time MILP, or the CP models Therefore, we propose a decomposition method to reduce the model size and decrease the computational expense.

One straightforward decomposition method is to decompose the problem block by block. We first conduct the optimization for each block to find optimal task sequences, and then regard the optimized task sequences as given, to make the allocations for the machines. This method divides the problem into a typical crane scheduling problem and a typical machine scheduling problem. However, there are two disadvantages of this straightforward decomposition method. First, even while considering cells in one block, the model size is still large. For example, if one cell needs four tasks, then ten cells will involve forty tasks, with a given number of hours within a specified time horizon. Second, limited to capacities of machines, there can be much difference between the optimal task sequences in each block and those across different blocks. Fixing task sequences in advance leaves limited flexibility for optimizing machine assignments.

Due to the limitations of the straightforward decomposition method, we propose an improved decomposition method, which decomposes the problem cell by cell. We first define three statuses for each cell: “completed”, “in operation”, and “to be operated”. Only the cells “in operation” will utilize the machines, so we do not need to consider the cells which have in status “completed”, or “to be operated” when assigning machines. In real cases, it is impossible that all cells are “in operation” at the same time. Therefore, with a maximum number of cells “in operation” status, when one cell changes the status from “in operation” into “completed” status, there will be another cell turning from “to be operated” into “in operation” status. This decomposition strategy is presented below in pseudocode with an illustrative example, involving some additional sets.

Additional sets

A_n: The set of current "in operation" cells in block n.

K_j: The set of subtasks belonging to cell j.

AK: The set of active tasks to be optimized.

CK: The set of completed tasks.

OK: The set of ongoing tasks, i.e. the object is being transferred.

Algorithm: Machine-based decomposition strategy

Assume: maximum m cells are at "in operation" status at a time
Initialize:
  Set $A_n \leftarrow \{j_1, j_2, \dots, j_m\} \subseteq J_n, \forall n \in N$
  Set $AK \leftarrow \bigcup_{\forall j \in A_n} K_j$
  Set $CK \leftarrow \emptyset$
  $PosI_c \leftarrow Pos_{c,0}, \forall c \in C^n$
while $AK \neq \emptyset$ **do**
  Optimize active tasks AK
  Record $t \leftarrow$ time when first cell c_1 across all blocks is completed
  Record $t_q\ (< t), \forall q \in Q \leftarrow$ last usage time before t
  Update:
    $PosI_c \leftarrow Pos_{c,t}, \forall c \in C^n$
    Machine availability at t_q
    The sets of A_n, AK, CK and OK
      $A_n \leftarrow A_n \backslash \{c_1\}$
      for $j \in A_n$ **do**
        for $k \in K_j$ **do**
          if k is completed **then**
            $CK \leftarrow CK \cup \{k\}$
            $AK \leftarrow AK \backslash \{k\}$
          else // the object is being transferred by crane c
            $POSp_{j,k} \leftarrow POS_{c,t}$
            $StartOf(Intv_{j,k}) \leftarrow 0$
  if $\exists j \in J_n \backslash A_n$ from the same block as c_1 **then**
    $A_n \leftarrow A_n \cup \{j\}$
    $AK \leftarrow AK \bigcup_j K_j$
end while // $|A_n| \leq m, \forall n \in N$

The algorithm above describes the machine-based decomposition strategy, and we give an illustrative example below for better explanation. In Fig.11, we assume in each block, a maximum of three cells are "in operation" at the same time. Therefore, we start with the optimization of six cells in block 1 and 2. After optimization, we find that cell 1 in block 1 is first completed. Then we update the crane positions and the subtasks completion status in the other five cells. Regardless of whether cell 4 in block 1 will be in operation next, we should add it in to the next optimization stage. Then in our second optimization, we find that the next completed cell is cell 1 in block 2. Therefore, when we conduct our third-time optimization iteration, we add cell 4 in block 2 to

ensure a maximum of three cells in operation in block 2. Such iterative optimizations can be performed until the number of the remaining incomplete cells are less equal than three in each block. The advantage of this decomposition method is that the scale of each small piece of optimization subproblem will not be so large, depending on the maximum number of cells operating at the same time. Furthermore, if the operations are disturbed due to unexpected events, we can record and update the positions of cranes and the status of subtasks of cells to apply a new optimization.

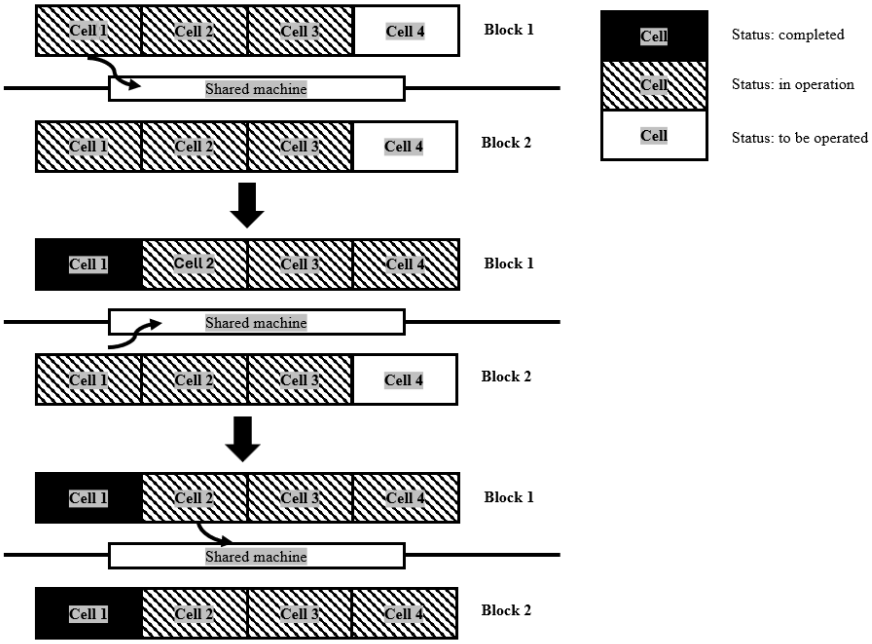

Fig.11. Decomposition strategy.

7. Results

7.1. Computational environment

All models are executed on a computer with an Intel Core i7-13620H CPU (2.40 GHz), 16 GB of RAM, and a 512 GB SSD. The MILP models are run in GAMS 46.5.0 using Gurobi 11.0 (https://www.gurobi.com.) as the solver. The constraint programming (CP) models are implemented in Python 3.12 using OR-Tools 9.8 with the CP-SAT (Perron L. and Didier F., 2025) solver. The time horizon is set to 30 minutes and 150 minutes in the small and medium examples, respectively. The time horizon is set to 100 minutes in the real-world examples in each iteration.

7.2. Small-scale examples

Example 1.1: We first consider a small-scale example for 2 cells with anode operations in single-crane conditions, which consists of 8 steps (Fig. 12). The details of the anode operations are presented in Section 4.2. Each step will be represented as a separate task in the following crane

scheduling examples. In this example, crane 1 moves between the left and the right boundaries to conduct anode operations for cells. The corresponding parameters are listed in Table 1.

Table 1. Parameters for Examples 1 and 2.

Pickup time (min)	1
Delivery time (min)	1
Velocity (m· min^{-1})	1
Safety distance (m)	1
Distance unit (m)	1
Cleaning time (min)	5
Stripping machine operation time (min)	3

We assume that cell 1 and cell 2 share the same position as the group. The relative positions of the group, the stripping machine, and the anode machine are shown in the figure, with one-meter intervals between them. Since we only consider cells in one block in this example, conflicts among the machines are neglected. All cells in one block share the same electrical power supply. Therefore, when one cell is under operation, the circuit is interrupted, and all other cells in the block are unable to operate. Our goal is to minimize the completion time for these two cells, i.e., inactive time. We apply discrete-time MILP, continuous-time MILP, and CP models to this small example and compare their performances. In addition, we compare the completion time under two strategies, operating the cells sequentially versus allowing cross-execution between cells.

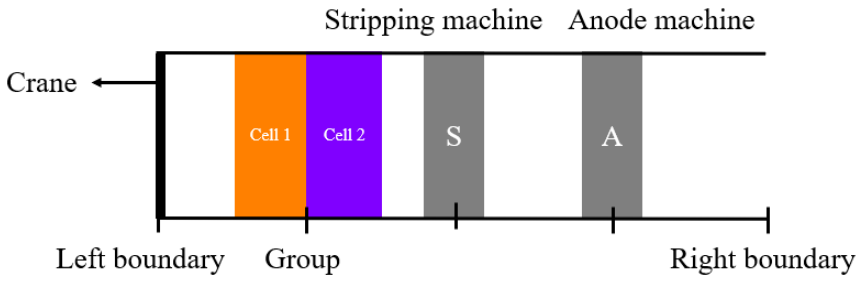

Fig.12. Crane scheduling diagram for example 1.1.

In this example, since there is only one crane, position interference does not occur during operations (Fig.13 (a) and (b)). Furthermore, instead of operating cell by cell, it is more efficient to regard the smallest unit as step for optimization. Each cell with anode operations requires 16 minutes to complete (Fig.13 (c)). If the crane processes the two cells sequentially without optimization, the total time would be 32 minutes. In contrast, the optimized solution, which allows overlapping operations between cells, completes both in just 30 minutes (Fig.13 (d)), resulting in a time saving of 6.25%.

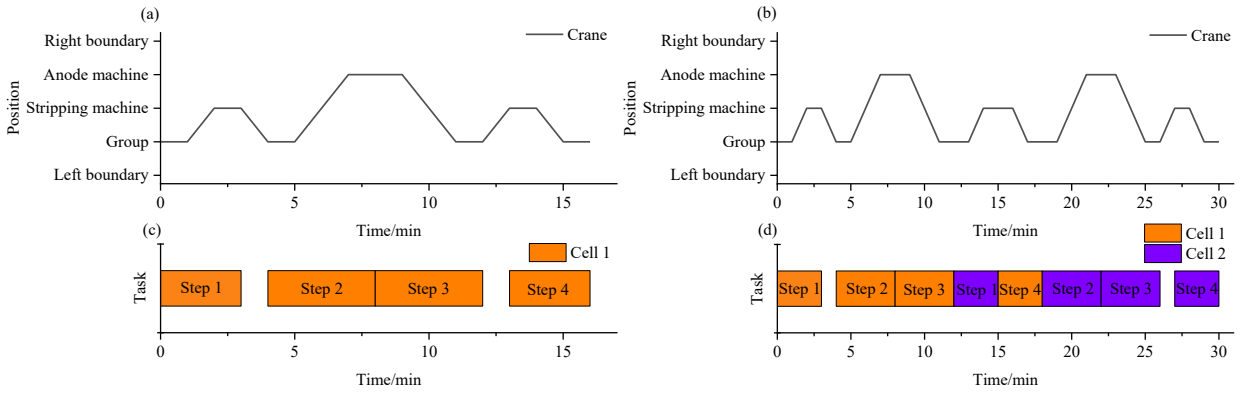

Fig.13. a) Crane's time-position figure for one cell, b) Crane's time-position figure for two cells, c) Crane's time-task figure for one cell, d) Crane's time-task figure for two cells.

The first column of Table 2 illustrates the details of the discrete-time MILP model. Apart from the objective variables, almost all variables in the models are binary variables. Furthermore, the model has a poor LP relaxation. As a result, it takes over 1 minute to solve the problem.

Since in block 1, there is only one crane for operation, we can apply the continuous-time MILP model without considering crane interference. For the continuous-time MILP model, in the second column of Table 2, the number of constraints is reduced from 10^5 to 10^2, and the CPU time is decreased from more than 1 minute to 0.02 seconds. Therefore, the continuous-time MILP models have better performance than the discrete-time MILP models in single-crane scheduling problems.

Given the weak LP relaxation of both MILP models, we turn to constraint programming (CP) models, which handle integer constraints more efficiently. In CP models (Table 2), all of the variables are integer variables, without any continuous variables. The constraint numbers as well as the CPU time imply that CP models perform as efficiently as the continuous-time MILP models.

Table 2. Model information for discrete-time MILP, continuous-time MILP, and CP models for one cell in Example 1.1.

	Discrete-time MILP	Continuous-time MILP	Constraint programming (CP)
Discrete variables	403	12	69
Continuous variables	2	46	0
Constraints	139,781	164	113
LP relaxation (min)	0.38	8	N/A
Objective value (min)	16	16	16
CPU time	62.23 s	0.02 s	0.02 s

Next, Table 3 reports the results of the three models for a slightly larger problem involving two cells. For the discrete-time MILP models, the constraint number is increased to 600,000 with only around 250-variable increase. This is mainly due to the added combinatorial complexity caused by the additional sequencing and assignment possibilities. Also, the CPU time rises significantly from 1 minute to 2 hours. The large increase in the number of constraints and the CPU time indicates

that for discrete-time MILP models, even a small addition of discrete times or discrete positions can lead to an exponential increase in model size. The continuous-time MILP model performs very well to solve the problem in 0.05 seconds (Table 3). Though there is a considerable increase in the number of variables and constraints in CP models, the computational time is only 0.02 seconds. This is because CP solvers are effective for handling large combinatorial spaces by applying domain reduction, constraint propagation, and efficient search strategies. In summary, for single-crane scheduling problems, the continuous-time MILP and the CP models both significantly outperform the discrete-time MILP models.

Table 3. Model information for discrete-time MILP, continuous-time MILP, and CP models for two cells in Example 1.1.

	Discrete-time MILP	Continuous-time MILP	Constraint programming (CP)
Discrete variables	651	56	233
Continuous variables	2	82	0
Constraints	628,569	583	417
LP relaxation (min)	0.33	8	N/A
Objective value (min)	30	30	30
CPU time	2.02 h	0.05 s	0.02 s

Example 1.2: We next present the same tasks as in the previous example, but this time with two cranes (Fig.14). We use the same values as the previous example 1.1 (Table 1). Additionally, since crane 1 and crane 2 share the same track as illustrated in Fig.14, a safety distance of 1 meter has to be always maintained between them. The objective is to minimize the completion time for two cells. Since the continuous-time MILP cannot be applied to the problem, we only compare the discrete-time MILP and CP models.

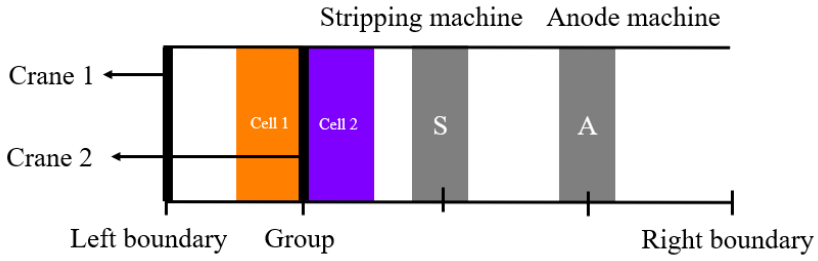

Fig.14. Crane scheduling diagram for example 1.2.

From Fig.15 (a) and (b), it is shown that crane 2 always stays on the right of crane 1, avoiding intersections during operations. As long as the paths do not cross, crane 1 and crane 2 can conduct operations at the same time. For example, in Fig.15 (a), crane 1 first picks up the spent steel sheet from group, and then moves left to leave space for crane 2 picking up spent copper anode from the group. Then, as crane 2 proceeds to the anode machine for delivery, crane 1 moves from the left boundary to the stripping machine to complete step 1. As a result, steps 1 and 2 are completed at the same time. In addition, under multi-crane conditions, similar to the single-crane case, allowing

multiple cells to operate simultaneously leads to a reduction in total completion time. Specifically, the time is shortened by 2 minutes, from 24 minutes to 22 minutes, representing an 8.33% decrease.

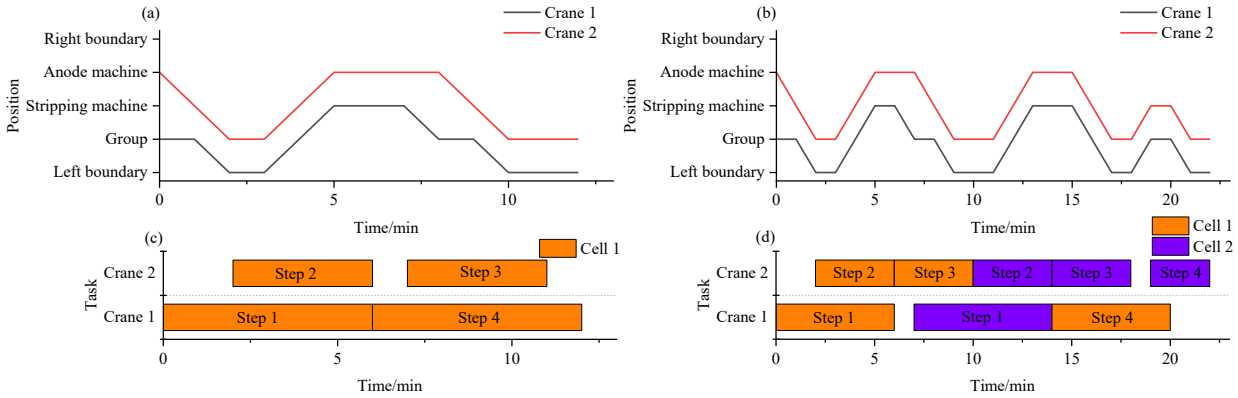

Fig.15 a) Crane's time-position figure for one cell, b) Crane's time-position figure for two cells, c) Crane's time-task figure for one cell, d) Crane's time-task figure for two cells.

Table 4. Solver information for discrete-time MILP, continuous-time MILP, and CP models for one cell in Example 1.2.

	Discrete-time MILP	Continuous-time MILP	Constraint programming (CP)
Discrete variables	806	Not applicable	1,499
Continuous variables	2		0
Constraints	275,826		2,801
LP relaxation (min)	0.17		N/A
Objective value (min)	12		12
CPU time (s)	80.06		0.50

Table 5. Model information for discrete-time MILP, continuous-time MILP, and CP models for two cells in Example 1.2.

	Discrete-time MILP	Continuous-time MILP	Constraint programming (CP)
Discrete variables	1,302	Not applicable	5,566
Continuous variables	2		0
Constraints	1,249,202		2,680
LP relaxation	0.17 min		N/A
Objective value	22 min		22 min
CPU time	2.96 h		1.46 s

Comparing Tables 4 and 5, the model size and computational time for the discrete-time MILP models increase exponentially with the number of tasks, resulting in poor scalability for multi-crane scheduling problems. The continuous-time MILP models are not applicable for multi-crane cases, as they fail to prevent crane interference. Among the three approaches, CP models perform the best, completing the optimization of two cranes handling 8 tasks in just 1.46 seconds. In summary, for multi-crane scheduling problems, CP models perform much better than discrete-time MILP models, while continuous-time MILP models are not applicable.

The results from example 1.1 and example 1.2 indicate that CP models are preferable for both single- and multi- crane scheduling problems. In the next case where we have a combination of single- and multi- crane scheduling problems, we apply CP models for the optimization.

7.3. Middle-scale examples

Example 2.1: Next, we consider a middle-scale example for a combined area with one crane in block 1 and two cranes in block 2, considering the conflict in the use of the machine between blocks as seen in Fig.16. The number of cells in each block is increased to four, conducting anode operations (as shown in Section 4.2), with a total of 32 tasks. Crane 1 can operate independently, while crane 2 and crane 3 must keep a safety distance of 1 meter between each other. Cells in block 1 and block 2 need to share the same stripping machine, and the stripping machine only has one slot to process the steel sheet with copper. Hence, non-conflict constraints must be implemented in this problem. We assume that once the crane places the item, the machine begins processing immediately, with no time delay in between. However, since the stripping machine can handle only one task at a time, a crane carrying a steel sheet with copper must wait above the machine if the machine is operating and can only proceed with unloading once the current task with the machine is completed. Based on actual operating procedures, block 1 and 2 begin their operations simultaneously. Therefore, the objective value is defined as the sum of completion times across both blocks. The minimum total completion time is identical to the minimum inactive time for cells. To solve this problem, we apply the constraint programming (CP) model in this example.

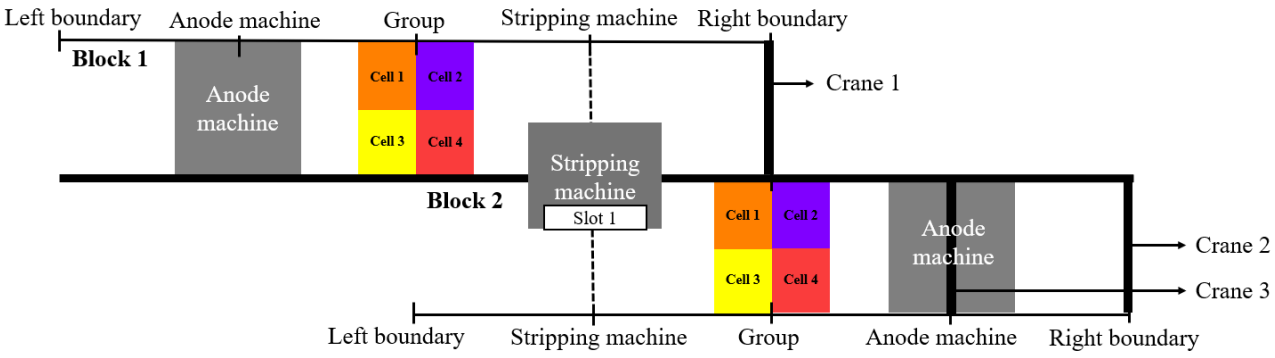

Fig.16. Crane scheduling diagram for example 2.1 and 2.2.

Fig.17 (a) and (b) illustrate the positions of the three cranes during operation. Crane 1 in block 1 operates independently, while crane 2 is positioned to the left of crane 3 in block 2. In this example, more focus is placed on the task schedules and the allocation of the stripping machine. In Fig.17 (c), the top section shows the task sequence for cells in block 1. The middle section illustrates the use of the stripping machine, indicating the assigned time slots and the corresponding cell operations from each block. The bottom section presents the crane assignments and task sequence for cells in block 2. The arrows in Fig.17 (c) represent the start time for the use of the machine. For example, the stripping machine will be in use after the delivery of the spent steel sheet, which is the end of step 1 in both cathode and anode operations. Therefore, after the end of the delivery

in step 1 for each cell, the machine spends 3 minutes processing it. In Fig.17 (c), it is shown that the stripping machine is used alternatively by blocks without conflicts. After optimization, the four cells in block 1 take 50 minutes, and the four cells in block 2 take 30 minutes, resulting in a total completion time of 80 minutes as the objective value.

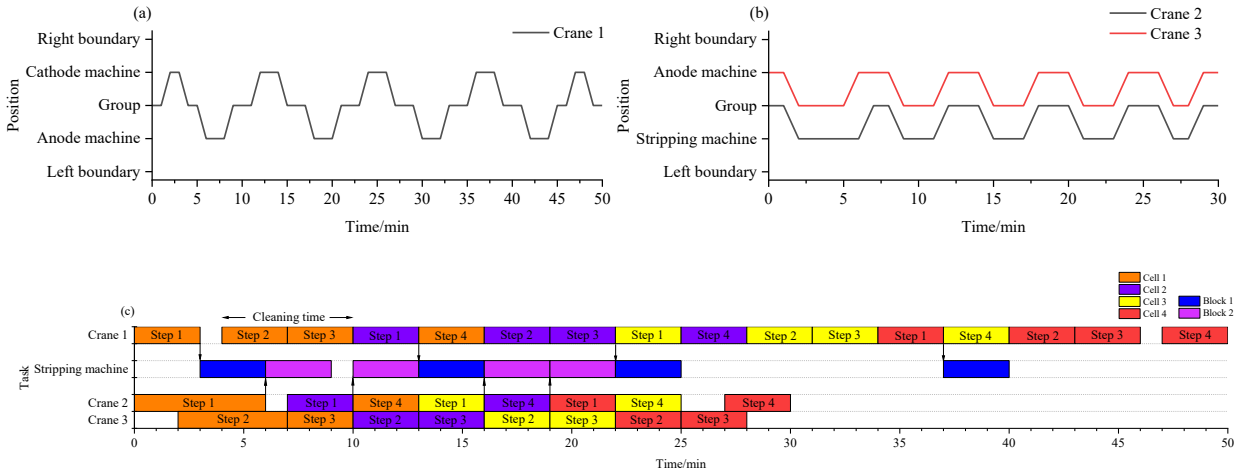

Fig.17. a) Time-position figure of crane 1 in block 1, b) Time-position figure of crane 2 and 3 in block 2, c) Time-task figures for all three cranes and the corresponding allocation for the use of the stripping machine.

This CP model has about 12,500 variables and 28,500 constraints, with a CPU time of 18.18 seconds as seen in instance 4 of Table 6. It is effective to take less than 20 seconds for the optimization of 4 cells with 24 tasks, under the operations of 3 cranes and 1 shared machine. However, when addressing larger problems, for example, increasing the number of cells per block to five, shown in instance 5 in Table 6, the computational time increases to more than 23 minutes.

Table 6 shows the results for instances in which each block contains from 1 to 5 cells, where B1 and B2 represent block 1 and 2, and 1-5 A means 1 to 5 cells with anode operations. As shown in the table, the average processing time per cell decreases as more cells are included in the optimization. This is because considering more cells increases the potential possibilities for the combinations of steps to reduce the total completion time. However, the CPU time grows exponentially with the number of cells as seen in Fig.18. To make the model computationally efficient in a large-scale problem, in the next sub-example 2.2, we propose a machine-based decomposition strategy and evaluate its performance by comparing it to the optimal solution obtained from a single full-scale model in example 2.1.

Table 6: Model information for the CP model in Example 2.1.

Instance	Cells	Variables	Constraints	Block 1 (min)	Block (min)	Objective (min)	Time per cell (min)	CPU time (s)
1	B1 1A B2 1A	3,430	7,012	14	10	24	12	0.87
2	B1 2A B2 2A	6,218	13,687	26	18	44	11	1.63
3	B1 3A B2 3A	9,265	20,848	38	24	62	10.34	3.43
4	B1 4A B2 4A	12,571	28,495	50	30	80	10	18.17
5	B1 5A B2 5A	16,136	36,628	62	36	98	9.8	1404.44

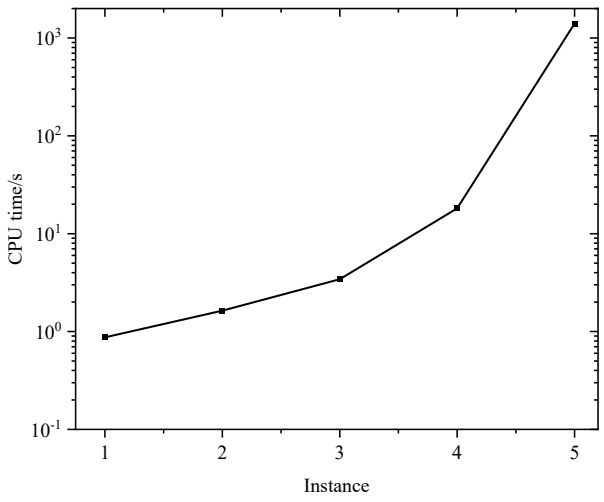

Fig.18. Computational times for different instances

Before introducing the decomposition method, we first use the optimal results as a reference. As shown in Fig.19, at most 2 cells in both block 1 and block 2 operate simultaneously. This implies that if additional cells are included in the optimization, the extra cells will not participate in operations.

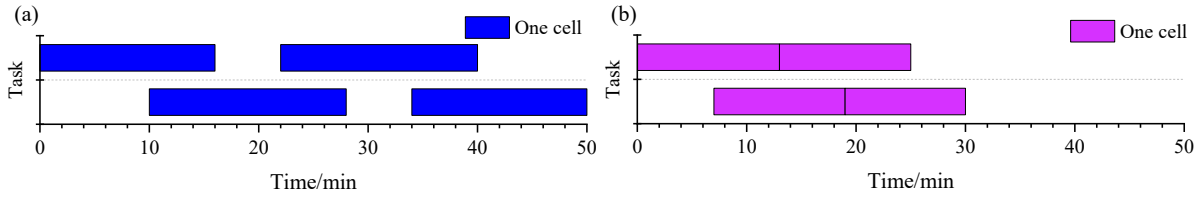

Fig.19. Operation times for cells in example 2.1 in a) block 1, b) block 2.

Example 2.2: After obtaining the optimal solution, we apply the decomposition method to the previous medium-size example to evaluate the performance of this solution strategy.

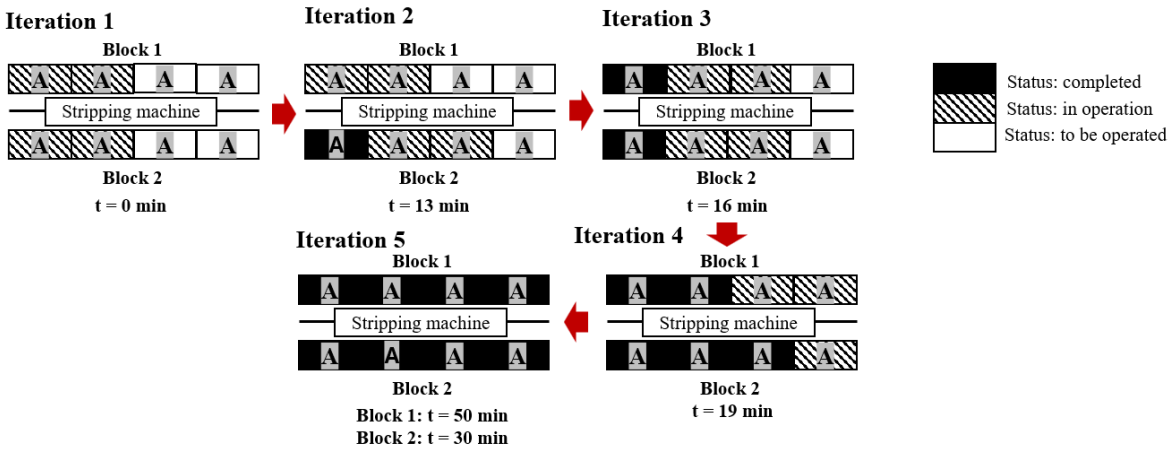

Fig.20. Decomposition strategy as example 2.2.

Fig.20 illustrates the iterations of the optimization process. As shown in the optimal solution in Fig.19, a maximum of 2 cells in blocks 1 and 2 are operated simultaneously. Based on this, example 2.2 applies the same upper limit when applying the decomposition strategy. As a result, the objective value through decomposition is consistent with that of the full-scale optimization as 80 minutes. The accumulated CPU time after decomposition is 4.81 seconds (Table 7), which is shorter than 18.17 seconds (Table 6 instance 4) for solving the full-size model. The advantage of the machine-based decomposition strategy is that it reduces the CPU time growth from exponential to linear (Fig.21). Another advantage is its greater flexibility in application. If crane operations are interrupted due to unexpected issues, the model can resume scheduling by inputting the current initial conditions for cranes and tasks. One possible limitation of this decomposition strategy is the difficulty in determining the maximum number of cells that can operate simultaneously. Unlike this medium-sized example, where the maximum number is identified from the optimal solution, in real-world conditions, this information is typically hard to obtain, which may lead to deviations from the true optimal performance.

Table 7. Solution details for each decomposition iteration in Example 2.2.

Iteration	Variables	Constraints	CPU time (s)	Accumulated CPU time (s)
1	6,218	13,687	1.634	1.63
2	4,885	10,478	1.203	2.84
3	3,593	7,328	0.806	3.64
4	4,885	10,479	0.98	4.62
5	1,585	2,452	0.184	4.81

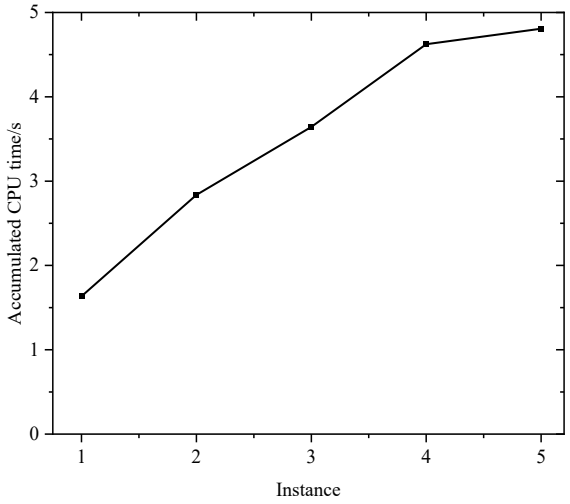

Fig.21. The accumulated computational times for each iteration

7.4. Large-scale examples

Finally, we consider large examples corresponding to real-word problems, which are described in the supplementary material and which show that a problem with 32 cells can be solved in less than one minute.

8. Conclusions

In this paper, we formulate and compare the performance of discrete-time MILP, continuous-time MILP, and CP models in both single- and multi-crane scheduling problems, considering time and space constraints including sequence of tasks, arrangement of cranes, non-interfering movements. As a result, for single-crane scheduling problems, the continuous-time MILP and the CP models perform better than the discrete-time MILP model. For multi-crane scheduling problems, the CP model outperforms the discrete-time MILP model, while the continuous-time MILP model is not applicable due to the failure in avoiding crane interference. In summary, the CP model is efficient in both single- and multi- crane scheduling problems.

To make it more efficient to solve large-scale problems, we propose a machine-based decomposition strategy for the CP model. It turns the computational time from exponential growth to linear growth with the increasing number of cells and tasks. Furthermore, it offers greater flexibility. In the condition of unexpected disruptions, the positions of cranes and the status of tasks can be recorded and updated, enabling re-optimization based on the latest system state.

The results of real-world case studies demonstrate the effectiveness of the proposed solution strategy. Given 32 cells with 128 tasks, it takes 34.02 seconds with 24 iterations to achieve an objective value of total completion time of 340 minutes.

While this work provides a useful optimization strategy for crane scheduling problems in tankhouse systems, it does not take into consideration uncertainties such as crane breakdowns and

machine shutdowns, as well as human factors, which could affect the cell cleaning times. This framework can be further expanded by incorporating stochastic programming techniques to account for operational uncertainties, thus making the solutions closer to real-world conditions. In addition, it would be interesting to explore the integration of energy consumption models into this crane scheduling framework, optimizing not only operational efficiency but also energy use.

Supplementary materials

Supplementary material associated with this article can be found in the attached document.

References

Agra, Agostinho, and Maryse Oliveira. 2018. "MIP Approaches for the Integrated Berth Allocation and Quay Crane Assignment and Scheduling Problem." *European Journal of Operational Research* 264 (1): 138–48. https://doi.org/10.1016/j.ejor.2017.05.040.

Aron, Ionut, Latife Genç-Kaya, Iiro Harjunkoski, Samid Hoda, and John N. Hooker. 2010. "Factory Crane Scheduling by Dynamic Programming." *Operations Research, Computing and Homeland Defense (ICS 2011 Proceedings)*, 93–107.

Bierwirth, Christian, and Frank Meisel. 2009. "A Fast Heuristic for Quay Crane Scheduling with Interference Constraints." *Journal of Scheduling* 12 (4): 345–60. https://doi.org/10.1007/s10951-009-0105-0.

Daganzo, Carlos F. 1989. "The Crane Scheduling Problem." *Transportation Research Part B: Methodological* 23 (3): 159–75. https://doi.org/10.1016/0191-2615(89)90001-5.

Emde, Simon. 2017. "Optimally Scheduling Interfering and Non-interfering Cranes." *Naval Research Logistics (NRL)* 64:476–89.

Feng, Kai, Lingzhi Yang, Dongfeng He, Shijing Lin, and Buxin Su. 2022. "A Study on Deep Reinforcement Learning-Based Crane Scheduling Model for Uncertainty Tasks," High Temperature Materials and Processes, 41 (1): 469–81. https://doi.org/10.1515/htmp-2022-0040.

Fuke, Tomonao, Masato Arimitsu, Hideaki Aoki, and Kazunori Tanisaki. 2025. "Prevention of Anode Passivation in Copper Electrorefining." In *Proceedings of the 63rd Conference of Metallurgists, COM 2024*, edited by Metallurgy and Materials Society of CIM, 337–39. Cham: Springer Nature Switzerland.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual. 2024. Available: https://www.gurobi.com.

H. -P. Hsu, C. -C. Chou, and C. -N. Wang. 2022. "Heuristic/Metaheuristic-Based Simulation Optimization Approaches for Integrated Scheduling of Yard Crane, Yard Truck, and Quay Crane Considering Import and Export Containers." *IEEE Access* 10:64650–70. https://doi.org/10.1109/ACCESS.2022.3180752.

Hooker, John N. 2002. "Logic, Optimization, and Constraint Programming." *INFORMS Journal on Computing* 14 (4): 295–321.

Kang Shih-Chung, Chi Hung-Lin, and Miranda Eduardo. 2009. "Three-Dimensional Simulation and Visualization of Crane Assisted Construction Erection Processes." *Journal of Computing in Civil Engineering* 23 (6): 363–71. https://doi.org/10.1061/(ASCE)0887-3801(2009)23:6(363).

Kizilay, Damla, and Deniz Türsel Eliiyi. 2021. "A Comprehensive Review of Quay Crane Scheduling, Yard Operations and Integrations Thereof in Container Terminals." *Flexible Services and Manufacturing Journal* 33 (1): 1–42. https://doi.org/10.1007/s10696-020-09385-5.

Li, Wenkai, Yong Wu, Mark Goh, Wenkai Li, Yong Wu, and Mark Goh. 2015. *A Continuous-Time Model for Multiple Yard Crane Scheduling with Last-Minute Job Arrivals*. Springer.

Li, Wenkai, Yong Wu, M.E.H. Petering, Mark Goh, and Robert de Souza. 2009. "Discrete Time Model and Algorithms for Container Yard Crane Scheduling." *European Journal of Operational Research* 198 (1): 165–72. https://doi.org/10.1016/j.ejor.2008.08.019.

Liu, Zhengchao, Liuyang Xu, Chunrong Pan, Xiangdong Gao, Wenqing Xiong, Hongtao Tang, and Deming Lei. 2023. "A Low-Carbon Scheduling Method of Flexible Manufacturing and

Crane Transportation Considering Multi-State Collaborative Configuration Based on Hybrid Differential Evolution." *Processes* 11 (9). https://doi.org/10.3390/pr11092737.

Meng, Leilei, Chaoyong Zhang, Biao Zhang, Kaizhou Gao, Yaping Ren, and Hongyan Sang. 2023. "MILP Modeling and Optimization of Multi-Objective Flexible Job Shop Scheduling Problem with Controllable Processing Times." *Swarm and Evolutionary Computation* 82 (October):101374. https://doi.org/10.1016/j.swevo.2023.101374.

Moccia, Luigi, Jean-François Cordeau, Manlio Gaudioso, and Gilbert Laporte. 2006. "A Branch-and-Cut Algorithm for the Quay Crane Scheduling Problem in A Container Terminal." *Naval Research Logistics (NRL)* 53 (February):45–59. https://doi.org/10.1002/nav.20121.

Naeem, Doaa, Mohamed Gheith, and Amr Eltawil. 2023. "A Comprehensive Review and Directions for Future Research on the Integrated Scheduling of Quay Cranes and Automated Guided Vehicles and Yard Cranes in Automated Container Terminals." *Computers & Industrial Engineering* 179 (May):109149. https://doi.org/10.1016/j.cie.2023.109149.

Peng, Gongzhuang, Youqi Wu, Chunjiang Zhang, and Weiming Shen. 2021. "Integrated Optimization of Storage Location Assignment and Crane Scheduling in an Unmanned Slab Yard." *Computers & Industrial Engineering* 161 (November):107623. https://doi.org/10.1016/j.cie.2021.107623.

Perron L., and Didier F. (2025, February 17). CP-SAT (v9.12). Google. Https://Developers.Google.Com/Optimization/Cp/Cp_solver/.

Pesant, Gilles. 2014. "A Constraint Programming Primer." *EURO Journal on Computational Optimization* 2 (3): 89–97. https://doi.org/10.1007/s13675-014-0026-3.

Peterson, Ben, Iiro Harjunkoski, Samid Hoda, and J.N. Hooker. 2014. "Scheduling Multiple Factory Cranes on a Common Track." *Computers & Operations Research* 48 (August):102–12. https://doi.org/10.1016/j.cor.2014.03.005.

Qin, Tianbao, Yuquan Du, Jiang Hang Chen, and Mei Sha. 2020. "Combining Mixed Integer Programming and Constraint Programming to Solve the Integrated Scheduling Problem of Container Handling Operations of a Single Vessel." *European Journal of Operational Research* 285 (3): 884–901.

Raman, R., and I.E. Grossmann. 1994. "Modelling and Computational Techniques for Logic Based Integer Programming." *An International Journal of Computer Applications in Chemical Engineering* 18 (7): 563–78. https://doi.org/10.1016/0098-1354(93)E0010-7.

Rossi, Francesca, Peter van Beek, and Toby Walsh. 2008. "Chapter 4 Constraint Programming." In *Foundations of Artificial Intelligence*, edited by Frank van Harmelen, Vladimir Lifschitz, and Bruce Porter, 3:181–211. Elsevier. https://doi.org/10.1016/S1574-6526(07)03004-0.

S. Grobbelaar, C. Dorfling, and M. Tadie. 2023. "Evaluating the Value of Including Physicochemical Property Correlations on the Prediction Capability of a Dynamic Electrowinning Model." In . The Southern African Institute of Mining and Metallurgy.

Siitari, Eero. 2022. "Crane's Information System in Operations Management."

Tan, Caimao, Wei Yan, and Jiantao Yue. 2021. "Quay Crane Scheduling in Automated Container Terminal for the Trade-off between Operation Efficiency and Energy Consumption." *Advanced Engineering Informatics* 48 (April):101285. https://doi.org/10.1016/j.aei.2021.101285.

Tanizaki, Takashi, Takayoshi Tamura, Hideaki Sakai, Yutaka Takahashi, and Taichi Imai. 2006. "A Heuristic Scheduling Algorithm for Steel Making Process with Crane Handling (< Special

Issue> Advanced Planning and Scheduling for Supply Chain Management).” *Journal of the Operations Research Society of Japan* 49 (3): 188–201.

Unsal, Ozgur, and Ceyda Oguz. 2013. “Constraint Programming Approach to Quay Crane Scheduling Problem.” *Transportation Research Part E: Logistics and Transportation Review* 59 (November):108–22. https://doi.org/10.1016/j.tre.2013.08.006.

Van Hentenryck, P., L. Michel, L. Perron, and J. -C. Régin. 1999. “Constraint Programming in OPL.” In *Principles and Practice of Declarative Programming*, edited by Gopalan Nadathur, 98–116. Berlin, Heidelberg: Springer Berlin Heidelberg.

Wang, Hongtao, Fulgencia Villa, Eva Vallada, and Rubén Ruiz. 2025. “Solving the Yard Crane Scheduling Problem with Dynamic Assignment of Input/Output Points.” *Computers & Operations Research* 173 (January):106853. https://doi.org/10.1016/j.cor.2024.106853.

Zhu, Aimin, Zhiqian Zhang, and Wei Pan. 2023. “Technologies, Levels and Directions of Crane-Lift Automation in Construction.” *Automation in Construction* 153 (September):104960. https://doi.org/10.1016/j.autcon.2023.104960.