IIE Transactions

ISSN: 0740-817X (Print) 1545-8830 (Online) Journal homepage: www.tandfonline.com/journals/uiie20

A new model and an efficient branch-and-bound solution for cyclic multi-hoist scheduling

Yun Jiang & Jiyin Liu

To cite this article: Yun Jiang & Jiyin Liu (2014) A new model and an efficient branch-and-bound solution for cyclic multi-hoist scheduling, IIE Transactions, 46:3, 249-262, DOI: 10.1080/0740817X.2012.762485

To link to this article: https://doi.org/10.1080/0740817X.2012.762485

IIE Transactions (2014) **46**, 249–266

ISSN: 0740-817X print / 1545-8830 online
DOI: 10.1080/0740817X.2012.762485

A new model and an efficient branch-and-bound solution for cyclic multi-hoist scheduling

YUN JIANG[1] and JIYIN LIU[2,*]

[1] *Alcatel-Lucent, Cambridge, UK*
E-mail: iedscjy@gmail.com
[2] *School of Business and Economics, Loughborough University, Leicestershire, LE11 3TU, UK*
E-mail: j.y.liu@lboro.ac.uk

Received October 2011 and accepted November 2012

This article studies the multi-hoist cyclic scheduling problem in electroplating lines where the processing time of parts in each tank must be within given lower and upper limits and part moves between tanks are allowed in both directions along the line. The problem arises in electroplating lines such as those used in the production of printed circuit boards and has previously been modeled as a mixed-integer linear program. The possible relative positions of any pair of moves are analyzed and a set of linear constraints is derived that expresses the no-collision requirements for hoists. Based on the analysis, a new mixed-integer linear programming model is formulated for the multi-hoist scheduling problem. An efficient branch-and-bound strategy is proposed to solve the problem. Computational results show that the new model can be solved much more quickly than an existing model in the literature and that the proposed solution method is more efficient in solving the problem than a commercial software package.

Keywords: Hoist scheduling, multiple hoists, integer programming, branch and bound

1. Introduction

Hoist scheduling problems arise in electroplating lines such as those for the treatment of Printed Circuit Boards (PCBs). An electroplating line consists of a number of tanks containing chemical solutions. In production, a number of PCBs are loaded into a carrier at a loading station to form a production unit load. The carrier (with the load of PCBs) is then processed in the tanks in a given sequence. At most one carrier can be processed in a tank at a time. The processing time of the carrier in each tank must take a value within a given window defined by a lower limit and an upper limit. After finishing all processing stages, the carrier is moved to an unloading station where the PCBs are unloaded.

Programmable hoists are mounted on a common track above the electroplating line, responsible for moving carriers among tanks (here the loading and unloading stations are also called tanks for convenience). The moves of carriers among tanks may be in either direction along the line. There is no buffer between tanks in the system. In addition, the moves of carriers among tanks must be performed as quickly as possible because long exposure in the air can cause oxidization of the PCBs and affect their quality. To perform a carrier move, a hoist picks the carrier up from a tank, moves it along the track to the tank of the next stage, and drops it off into that tank. In practice, the time for picking up or dropping off a carrier is a significant portion of the total time for the carrier move. Therefore, it is necessary to consider the picking up and dropping off times explicitly and separately from the travel time in defining and solving a multi-hoist scheduling problem because the exact position of each hoist must be known at any time during the period of performing a move in order to avoid collisions between hoists.

The electroplating line processes PCBs in batches and a batch is made up of a large number of identical PCBs. When a batch is finished, the line needs to be adjusted (e.g., replacing the solutions in the tanks or adjusting their concentration) and the hoists need to be programmed to perform a new sequence of transfer operations repeatedly according to the process requirements of the new batch. The period between two successive repetitions of hoist operations is a production cycle. Although processing times are allowed to take any values within their limits, they must be determined before starting the batch because the concentration of the solutions in the tanks needs to be adjusted according to the chosen processing times. In addition, to

*Corresponding author

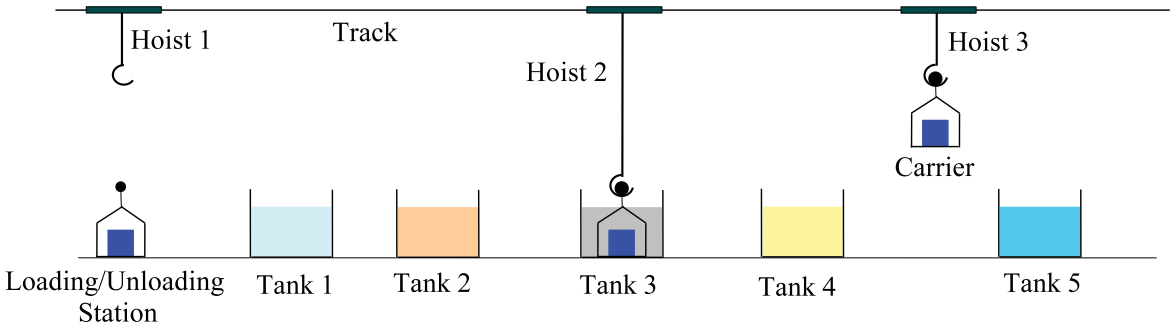

Fig. 1. An illustrative electroplating line with five tanks and three hoists (color figure provided online).

ensure consistent product quality, the processing time in a tank must be the same for all PCBs in the batch. An illustrative example line consisting of five tanks and three hoists is shown in Fig. 1.

An optimal schedule of hoist operations is difficult to obtain, especially when there are multiple hoists that must not collide with each other on the track. In practice, schedules are constructed manually and it takes several days to construct one. While the manual scheduling results, even by experienced engineers, are often sub-optimal, the long lead time also makes the process inflexible. In today's highly competitive electronics market, electronics manufacturers require PCBs supplied in small batch sizes and with a short lead time. Therefore, a computer tool to help make optimal hoist schedules is necessary considering the market requirement and to achieve higher productivity of the lines.

Hoist scheduling problems have attracted attention from academic researchers for over 30 years since Phillips and Unger (1976) presented a mixed-integer programming model for a single-hoist problem. For work on hoist scheduling problems before 2002, please refer to Manier and Bloch (2003). They classified hoist scheduling problems into different categories and gave a detailed literature review. Research on a broader range of cyclic scheduling problems in robotic flowshops was reviewed by Crama *et al.* (2000). They included problems of scheduling multiple part types. However, most studies reviewed consider systems with only one material-handling robot. For systems with multiple robots, the robots were allowed to cross each other without considering collisions.

Most previous research on hoist scheduling has focused on the single-hoist case. The single-hoist scheduling problem and its variants have been proved NP-complete (Lei and Wang, 1989; Serafini and Ukovich, 1989; Crama and Van de Klundert, 1997). However, Liu *et al.* (2002) showed that practical single-hoist problems can be solved successfully because the number of tanks in a line with a single hoist rarely exceeds 15 in practice. For lines with more tanks, two or more hoists are usually used to avoid hoists becoming a bottleneck in the operation. Compared with the single-hoist problem, the multi-hoist problem is much more challenging because of the additional need to avoid collisions between the hoists. Research on multi-hoist scheduling problems has become active in recent years.

Lei and Wang (1991) were the first to study a multi-hoist problem, in which two hoists are considered and the carrier moves are all in one direction, from a tank to the adjacent tank on its right. By partitioning the line into two non-overlapping zones and assigning one hoist for each zone, the problem can be transformed to two single-hoist problems. Che and Chu (2004) proposed a branch-and-bound algorithm for a multi-hoist electroplating line in which all moves are in one direction. Leung *et al.* (2004) developed a mixed-integer programming model for a multi-hoist line which again considered moves all in one direction. Leung and Zhang (2003) extended the model to the general case in which moves are allowed in both directions. Zhou and Liu (2008) proposed a heuristic search algorithm, with a linear programming model embedded, for the two-hoist problem allowing carrier moves in both directions. Lei *et al.* (1993) and Armstrong *et al.* (1996) proposed heuristics for minimizing the number of hoists in electroplating lines, a problem related to multi-hoist scheduling. The problem we study here is the general one for which a model was previously formulated by Leung and Zhang (2003).

When the processing times are fixed constants rather than flexible within limits, the hoist scheduling problem becomes a no-wait version and can be solved in polynomial time. Kats and Levner (1997b) developed a polynomial solution to the no-wait single-robot scheduling problem. Kats and Levner (1997a) provided a polynomial procedure to determine the minimum number of robots needed in a cyclic no-wait flowshop for any given cycle length, improving an earlier work by Karzanov and Livshits (1978). Their procedure was also used to minimize the cycle length when the number of robots was given. However, the robots were considered independent and thus could cross each other without collision. Liu and Jiang (2005) proposed an optimal polynomial algorithm for the no-wait hoist scheduling problem with two hoists on a common track and no restrictions on move directions. Jiang and Liu (2007) transformed the no-wait hoist scheduling problem with an arbitrary number of hoists on a common track to a series of

shortest path problems and solved it in polynomial time. Leung and Levner (2006) presented an efficent algorithm for determining the minimum number of hoists required for all possible cycle times in a no-wait production process. Given the number of hoists, the algorithm could also find the minimum-time cyclic hoist schedule.

In this article, we study the general cyclic multi-hoist scheduling problem. Following most previous research and industrial practice, we consider the one-degree cycle in which one part unit enters the system in every cycle, though multi-degree cycles may further improve the system throughput (e.g., Che and Chu (2005)). In the rest of this article, we first give a detailed problem definition in Section 2. The no-collision requirements between hoists are then analyzed and transformed to some compact linear constraints in Section 3. Incorporating these constraints, a new mixed-integer programming model is then developed in Section 4. Section 5 presents a branch-and-bound algorithm that takes advantage of the problem structure. Computational results are reported in Section 6. Finally, conclusions are drawn in Section 7.

2. Problem definition

The problem studied here can be defined similar to Jiang and Liu (2007) except that the processing times here are flexible within their limits, whereas those in Jiang and Liu (2007) are fixed. Nevertheless, we still give the detailed problem statement here for completeness. Consider an electroplating line that consists of a loading station, n chemical tanks, and an unloading station. The tanks/stations are arranged in a line and labeled from left to right as tank 0 (loading station), tank 1, 2, ..., n, and tank $n+1$ (unloading station). The position of the loading station is defined as 0 and the distance between the loading station and tank i as w_i. For many lines, the loading station and the unloading station are the same physical station on the left end of the electroplating line and hence $w_{n+1} = 0$. Above the tanks, there are m hoists mounted on a common track. The hoists are labeled as hoists 1, 2, ..., m, from left to right. The left-most position that hoist 1 can reach is $w_l \leq 0$ and the right-most position that hoist m can reach is $w_r \geq \max\{w_i | i = 0, 1, \ldots, n+1\}$.

The operation of the line is cyclic. A carrier is introduced to the system in every cycle. Each carrier is first loaded with PCBs at the loading station and then processed in the chemical tanks one by one according to a given sequence $s_1, \ldots, s_n$ and finally unloaded at the unloading station. s_i is the tank number for the ith processing stage. The complete sequence of the tanks visited by each carrier is then $s_0, s_1, \ldots, s_n, s_{n+1}$, where s_0 ($\equiv 0$) and s_{n+1} are the loading and unloading stations, respectively. The processing time for the ith stage can be any value between a lower limit a_i and an upper limit b_i. In each tank at most one carrier can be processed at any time. The processing time for the same stage must be the same for all carriers due to the cyclic requirement and for consistency in quality.

When a carrier finishes its processing in tank s_i, a hoist must be available to pick it up, move it along the track, and then drop it off to tank s_{i+1}. In the rest of this article, the whole operation for a hoist to transfer a carrier from tank s_i to tank s_{i+1} is called move i and is denoted as m_i, $i = 0, 1, \ldots, n$. The times for picking up the carrier at the beginning of move i and for dropping off the carrier toward the end of move i are μ_i and η_i, respectively. The travel speed of a hoist holding a carrier is v. A hoist can perform at most one move at a time. After completing a move, the hoist is free to travel to the start point of another move to perform it. The maximum speed for a hoist travelling empty is λ ($\lambda \geq v$). In addition, to avoid collision, any pair of adjacent hoists must keep at least a safety distance d from each other. For convenience, we denote r_i to be the total time of move i. That is,

$$r_i = \mu_i + |w_{s_{i+1}} - w_{s_i}|/v + \eta_i, \qquad i = 0, 1, \ldots, n.$$

In cyclic operation, one carrier enters the line, another carrier leaves the line, and each of the moves $0, 1, \ldots, n$, is performed exactly once in every cycle. Without loss of generality, we define the beginning of a cycle as the starting time of move 0. The cycle length T is the duration between the starting time of m_0 for one carrier and that for the next. The move 0s of all carriers always begin at time point kTs where $k \in Z$. The moves in a cycle may correspond to different carriers. The hoists need to perform a series of operations to finish all the moves $0, 1, \ldots, n$, for one cycle and repeat the operations every cycle. The cyclic multi-hoist scheduling problem is then to schedule the hoists to perform all the moves in a cycle, under the tank and hoist constraints and non-collision requirements, so that the cycle length T is minimized (effectively the production throughput will be maximized).

The resulting schedule of the problem can be clearly presented as a time-way diagram. Figure 2 is an illustrative time-way diagram showing a schedule for an electroplating line with three tanks and two hoists. In such a diagram, the horizontal and vertical axes represent time and position along the line, respectively. A move of a carrier between two tanks is represented by three thick solid line segments, in which the sloped segment in the middle indicates that the carrier is being moved from a tank to another and the two horizontal segments at both ends correspond to the picking up and dropping off operations. Dashed thick lines in the diagram represent the operations of empty hoists.

The relationship between the moves of one carrier and the moves in a cycle can also be seen in the time-way diagram. For the example in Fig. 2, the cycle length $T = 70$. Two complete cycles are shown and two carriers enter the line at the beginning of the two cycles, respectively. Consider the carrier entering the line at time 0. It enters tank 2 at time 20 for the first-stage processing until time 50. Then it is moved from tank 2 to tank 3 for the second-stage

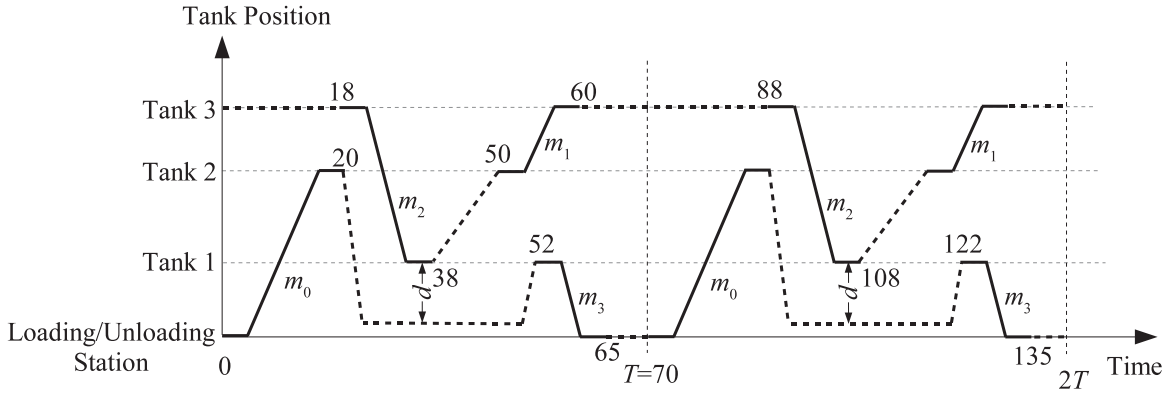

Fig. 2. Time-way diagram.

processing, which is from time 60 to time 88. The carrier leaves tank 3 at time 88 and enters tank 1 at time 108 for the third-stage processing until time 122. After finishing in tank 1, it returns to the loading/unloading station at time 135. Corresponding to this carrier, moves 0 and 1 are in the cycle $[0, T)$ and the moves 2 and 3 are in the cycle $[T, 2T)$. The moves 2 and 3 in the cycle $[0, T)$ are corresponding to the carrier entering the line at time $-T$. The starting times of moves 2 and 3 in the cycle $[0, T)$ are $88 - T = 18$ and $122 - T = 52$, respectively. In each cycle, hoist 1 is programmed to perform move 0 first and then move 3. Hoist 2 performs move 2 first and then move 1. The solid and dashed thick lines for a hoist linking together form the route of that hoist in the time-way diagram. To avoid collision between hoists, the routes of two adjacent hoists in the time-way diagram must keep at least the safety distance d vertically at any time.

To avoid splashing of the liquids in the processing tanks, the lifting up and dropping off of carriers cannot be very quick. As a result, the time to lift up or drop off a carrier can be significant in the total time of a move of the carrier. Therefore, it is necessary to represent a move as a three-segment shape in the time-way diagram. Phillips and Unger (1976) described the three parts of a move when presenting their single-hoist benchmark problem. Note that in the single-hoist case, the feasibility of a schedule will not change if each move is approximately represented as a straight line between the start point and the end point of the three-segment shape of the move. However, for the multi-hoist case, such an approximation may misinterpret some infeasible solutions as feasible and *vice versa*. For example, Fig. 3 shows some required moves for a multi-hoist problem in a time-way diagram. The approximation of a move is shown as a thin straight line between the start and the end of the move. From the diagram we can see that moves m_i and m_j in the figure cross each other and thus cannot have feasible hoist assignments, while they could be mistakenly assigned to two hoists if they were approximated as straight lines. On the other hand, moves m_h and m_k can be feasibly performed by two hoists while the straight line approximation would consider them crossing each other and thus infeasible. Previous studies often use the straight line approximation for moves. The new model in this article uses the three-segment representation for moves to ensure practical feasibility of its solution.

3. No-collision constraints

One challenge of multi-hoist scheduling is to avoid collisions between hoists. To present the constraints for the no-collision requirements, we define a decision variable t_i to represent the starting time of move i in cycle $[0, T)$. We consider that a move is in cycle $[0, T)$ if its starting time is within $[0, T)$. In this section we transform the no-collision requirements to constraints between each pair of t_i and t_j.

Jiang and Liu (2007) considered the no-wait version of the multi-hoist problem, in which the processing times and the transfer times are all fixed and therefore the t_i values are fixed for any given cycle length. They identified two types of no-collision constraints by considering individual moves and move-pairs, respectively, and proved that there is a feasible hoist schedule (each hoist has a feasible route and there is no collision between them) to perform all the moves if and only if both individual-move constraints and move-pair constraints are satisfied. For our general multi-hoist

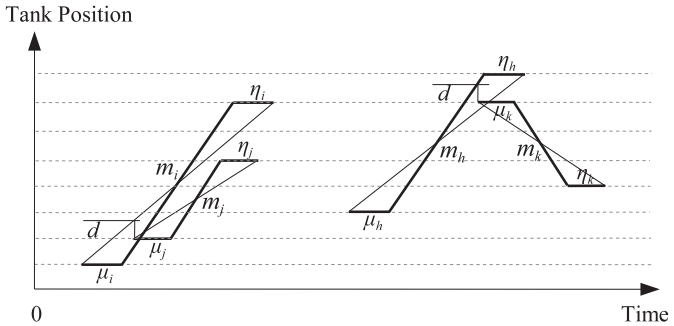

Fig. 3. Impact of move-shape on feasibility.

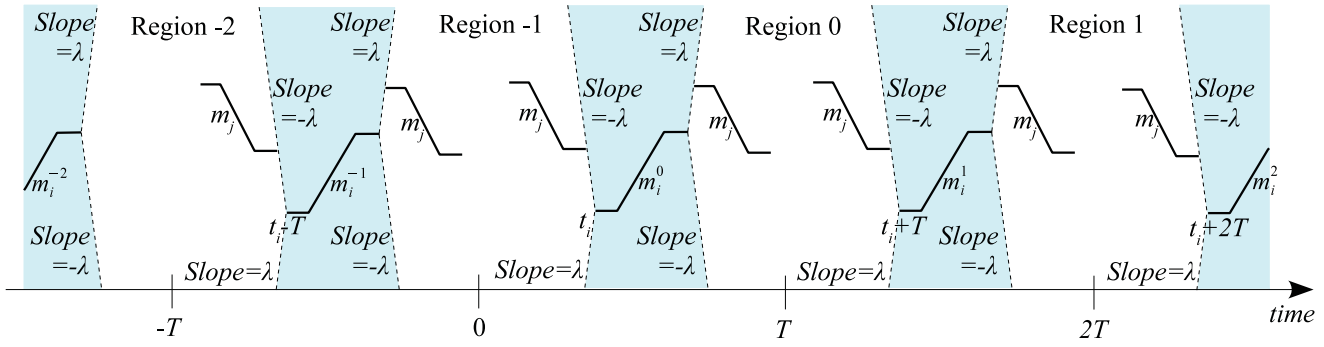

Fig. 4. Constraint for the situation where $h = 0$ (color figure provided online).

problem with processing time windows, the feasibility conditions still apply but the t_i are variables in the constraints.

3.1. *Individual-move constraints*

The individual-move constraints limit possible hoists to perform each move by considering the running ranges of the hoists on the track. Consider the safety distance d between any adjacent hoists, the left-most position that a hoist k can reach is $w_l + (k-1)d$ because at least a distance $(k-1)d$ must be kept between hoist 1 and hoist k. Similarly, at least a distance of $(m-k)d$ must be kept between hoist k and hoist m and the right-most position that hoist k can reach is $w_r - (m-k)d$. Therefore, hoist k can run only between positions $w_l + (k-1)d$ and $w_r - (m-k)d$ on the track. Clearly, a move i can be performed by a hoist k only if the starting and ending tanks of the move are both within the running range of the hoist. This can be expressed as

$$\min\left(w_{s_i}, w_{s_{i+1}}\right) \geq w_l + (k-1)d, \tag{1}$$

and

$$\max\left(w_{s_i}, w_{s_{i+1}}\right) \leq w_r - (m-k)d. \tag{2}$$

Considering that k is an integer and $1 \leq k \leq m$, inequalities (1) and (2) can be transformed to the following constraint, which limits the hoists that can perform move i:

$$\begin{aligned}\max\left\{1, \left\lceil m - \left(w_r - \max\left(w_{s_i}, w_{s_{i+1}}\right)\right)/d\right\rceil\right\} \leq k \\ \leq \min\left\{m, \left\lfloor\left(1 + \min\left(w_{s_i}, w_{s_{i+1}}\right) - w_l\right)/d\right\rfloor\right\}\end{aligned} \tag{3}$$

3.2. *Move-pair constraints*

Now consider a pair of moves i and j and denote the hoists performing them as hoists p and q, respectively. Define $h = q - p$. Then $h = 0$ means that p and q are the same hoist; $h > 0$ indicates that hoist p is on the left of q in the electroplating line (p is below q on the time-way diagram); and $h < 0$ indicates that p is on the right of q in the line (p is above q on the time-way diagram). The no-collision requirements restrict the relative starting times of moves i and j to avoid collision between hoists p and q. The constraints take different forms for different relative positions of the tanks involved in the move pair.

We discuss the constraints below using the case with the tank positions satisfying $w_{s_i} + hd \leq w_{s_{j+1}} < w_{s_{i+1}} + hd \leq w_{s_j}$ as an example.

For $h = 0$, the two moves are performed by the same hoist and there must be sufficient time for the hoist to travel empty between these two moves. Figure 4 shows move i in every cycle in a time-way digram. To distinguish them, we label the move i in the cycle $[kT, (k+1)T)$ as m_i^k in the figure. When performing a move i, the position of the hoist overlaps with the move. To come to the start point of the move before performing it, and to leave the end point of the move after performing it, the maximum speed the hoist can travel is λ. Considering this, a shaded area around each move i is shown in the figure as a prohibited region for move j. If any part of move j is in a prohibited region, there will not be enough time for the hoist to travel empty between the two moves. Between two adjacent prohibited regions is a non-prohibited region. The non-prohibited region between m_i^k and m_i^{k+1} is denoted as region k as marked in the figure. In a feasible schedule, move j must be in the non-prohibited regions. It can be seen from Fig. 4 that for the move j in region k, the difference between the starting times of moves i and j must satisfy the following constraint:

$$\begin{aligned}\left[kT + r_i + \left(w_{s_j} - w_{s_{i+1}}\right)/\lambda \leq t_j - t_i\right. \\ \left. \leq (k+1)T - r_j - \left(w_{s_i} - w_{s_{j+1}}\right)/\lambda\right].\end{aligned}$$

The lower limit in the above constraint corresponds to the situation where the starting point of move j touches the left boundary of the non-prohibited region, while the upper limits corresponds to the situation where the end point of move j touches the right boundary of the non-prohibited region.

Considering all of the non-prohibited regions, we have (for $h = 0$):

$$\begin{aligned}t_j - t_i \in \bigcup_{k=-\infty}^{\infty}\left[kT + r_i + \left(w_{s_j} - w_{s_{i+1}}\right)/\lambda,\right. \\ \left.(k+1)T - r_j - \left(w_{s_i} - w_{s_{j+1}}\right)/\lambda\right].\end{aligned}$$

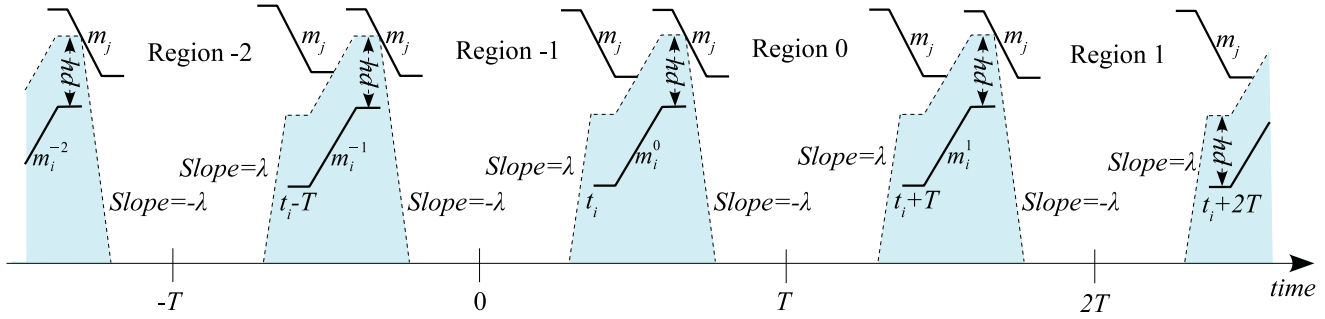

Fig. 5. Constraint for the situation where $h > 0$ (color figure provided online).

If $h > 0$, hoist q is above p on the time-way diagram as shown in Fig. 5. To avoid collision and allow maximum freedom for hoist q, hoist p keeps its position as low as possible: before performing move i, hoist p stays at the lowest position possible (as close to the left-most position in the system as possible) and then travels with maximum speed λ to tank s_i; the position of hoist p overlaps with move i during the period of performing the move; after finishing the move, hoist p travels with maximum speed to the lowest position possible. To avoid collision, hoist q, and hence move j, has to always keep a vertical distance of hd above the position of hoist p in the diagram. Reflecting this requirement, the shaded area around each move i in the figure represents a prohibited region for move j. The collision avoidance requirement restricts move j to be in one of the non-prohibited regions marked in Fig. 5. For the move j in region k, we have

$$kT + r_i - \mu_j - \left[w_{s_j} - \left(w_{s_{i+1}} + hd\right)\right]/v \le t_j - t_i \le (k+1)T + \mu_i + \left[w_{s_{j+1}} - \left(w_{s_i} + hd\right)\right]/v - r_j.$$

Considering all regions, the constraint (for $h > 0$) can be written as

$$t_j - t_i \in \bigcup_{k=-\infty}^{\infty} \left[kT + r_i - \mu_j - \left[w_{s_j} - \left(w_{s_{i+1}} + hd\right)\right]/v, (k+1)T + \mu_i + \left[w_{s_{j+1}} - \left(w_{s_i} + hd\right)\right]/v - r_j\right].$$

Similarly for $h < 0$, hoist q is below p on the time-way diagram as shown in Fig 6. For the move j in region k, we have

$$kT + r_i + \left[w_{s_j} - \left(w_{s_{i+1}} + hd\right)\right]/\lambda \le t_j - t_i \le (k+1)T - \left[w_{s_{j+1}} - \left(w_{s_i} + hd\right)\right]/\lambda - r_j.$$

Again because the m_j must be in the non-prohibited regions to avoid collision, we can obtain the following constraint (for $h < 0$):

$$t_j - t_i \in \bigcup_{k=-\infty}^{\infty} \left[kT + r_i + \left[w_{s_j} - \left(w_{s_{i+1}} + hd\right)\right]/\lambda, (k+1)T - \left[w_{s_{j+1}} - \left(w_{s_i} + hd\right)\right]/\lambda - r_j\right].$$

Note that the use of prohibited regions in time-way diagrams can be found in previous studies; e.g., Heinrichs and Moll (1997) and Leung and Zhang (2003). Both used straight line representation for moves in their diagrams. Note also that Heinrichs and Moll (1997) used the vertical axis for time and horizontal axis for position, whereas Leung and Zhang (2003) did not consider safety distance. Considering the practical shape of moves, the prohibited regions in this article become more complicated for situations with $h > 0$ and $h < 0$. For $h = 0$, the move-pair constraints are actually not for avoiding collisions between hoists. Rather, they are to avoid time conflict when two moves are being performed by the same hoist.

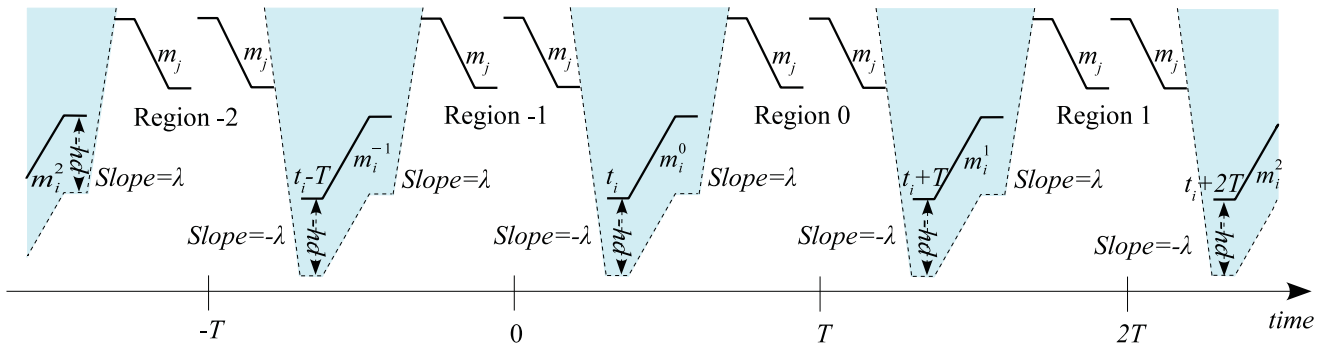

Fig. 6. Constraint for the situation where $h < 0$ (color figure provided online).

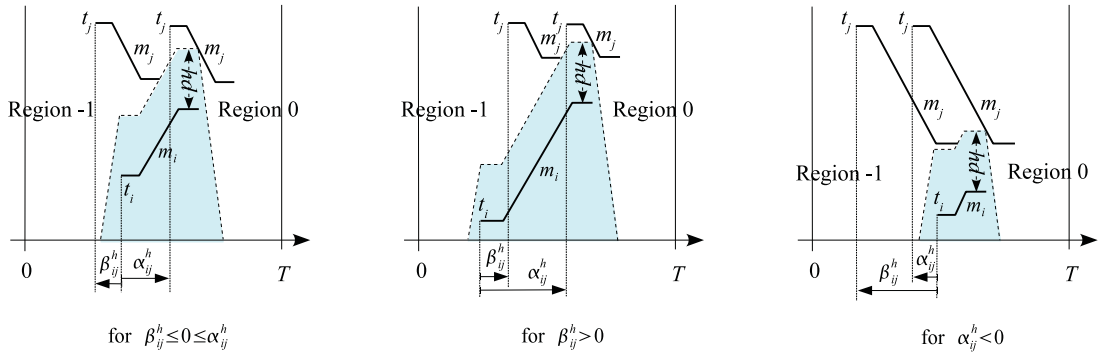

Fig. 7. Illustration of situations with different α_{ij}^h and β_{ij}^h values (color figure provided online).

Nevertheless, we may still refer to these constraints as no-collision constraints or collision-avoidance constraints in line with the situations where $h > 0$ and $h < 0$.

From the above discussions, we can see that in all situations the no-collision constraints for a pair of moves i and j always take the following general format:

$$t_j - t_i \in \bigcup_{k=-\infty}^{\infty} \left[kT + \alpha_{ij}^h, (k+1)T + \beta_{ij}^h\right], \tag{4}$$

where $h = q - p$, and hoists p and q are the hoists for performing moves i and j, respectively. α_{ij}^h is the lower limit for the time interval from the starting time of move i to the starting time of move j in the non-prohibited region on the right of move i, and β_{ij}^h is the upper limit for the time interval from the starting time of move i and the starting time of move j in the non-prohibited region on the left of move i, to avoid collision.

In Equation (4), α_{ij}^h and β_{ij}^h are different with the sign of h. In addition, α_{ij}^h and β_{ij}^h are dependent on relations among w_{s_i}, $w_{s_{i+1}}$, w_{s_j}, and $w_{s_{j+1}}$ when $h \neq 0$. The above discussion uses the case with $w_{s_i} + hd \le w_{s_{j+1}} < w_{s_{i+1}} + hd \le w_{s_j}$ as an example. A complete list of all possible cases and the corresponding formulae for α_{ij}^h and β_{ij}^h are presented in the Appendix. Note that α_{ij}^h equals the starting time of the m_j that touches the prohibited boundary after m_i minus the starting time of m_i, and β_{ij}^h equals the starting time of the m_j that touches the prohibited boundary before m_i minus the starting time of m_i. Therefore, we always have the following relationship:

$$-T \le \beta_{ij}^h < \alpha_{ij}^h \le T. \tag{5}$$

Different possible signs of α_{ij}^h and β_{ij}^h lead to three possible situations as illustrated in Fig. 7: $\beta_{ij}^h \le 0 \le \alpha_{ij}^h$, $\beta_{ij}^h > 0$, and $\alpha_{ij}^h < 0$.

Because $t_i \in [0, T)$, we have

$$t_j - t_i \in (-T, T). \tag{6}$$

Considering Equations (4) to (6), we can summarize the move-pair constraints for the above-mentioned three different situations as follows:

if $\beta_{ij}^h \le 0 \le \alpha_{ij}^h$,
$\alpha_{ij}^h - T \le t_j - t_i \le \beta_{ij}^h$ or $\alpha_{ij}^h \le t_j - t_i \le T + \beta_{ij}^h$;

if $\beta_{ij}^h > 0$,
$t_j - t_i \le \beta_{ij}^h - T$ or $\alpha_{ij}^h - T \le t_j - t_i \le \beta_{ij}^h$ or $t_j - t_i \ge \alpha_{ij}^h$;

if $\alpha_{ij}^h < 0$,
$t_j - t_i \le \beta_{ij}^h$ or $\alpha_{ij}^h \le t_j - t_i \le T + \beta_{ij}^h$ or $t_j - t_i \ge T + \alpha_{ij}^h$.

4. A new mixed-integer programming model

For convenience of modeling, we first introduce a few parameters that will be used together with those defined in Section 2. The individual-move constraint, Constraint (3), restricts the hoists that could be assigned to each move i. We define parameters l_i and u_i to represent this restriction.

l_i: the left-most hoist that could be assigned to move i restricted by the individual-move constraint: $l_i = \max\{m - \lfloor (w_r - \max\{w_{s_i}, w_{s_{i+1}}\})/d \rfloor, 1\}$.
u_i: the right-most hoist that could be assigned to move i restricted by the individual-move constraint: $u_i = \min\{1 + \lfloor (\min\{w_{s_i}, w_{s_{i+1}}\} - w_l)/d \rfloor, m\}$.
M: a large positive number.

To model the problem, we define the following decision variables:

T = the cycle length.
t_i = the starting time of move i in the cycle $[0, T)$, when a carrier finishes the ith-stage processing.

$$z_{ik} = \begin{cases} 1 & \text{if } m_i \text{ is performed by hoist } k \\ 0 & \text{otherwise} \end{cases}$$
$$i = 0, 1, \ldots, n, k = l_i, l_i + 1, \ldots, u_i.$$

$$x_{ij} = \begin{cases} 1 & \text{if } t_j - t_i \geq \alpha_{ij}^h \text{ is in effect} \\ 0 & \text{if } t_j - t_i \leq \beta_{ij}^h \text{ is in effect} \\ & \qquad i = 0, \ldots, n-1;\ j = i+1, \ldots, n. \end{cases}$$

$$y_{ij} = \begin{cases} 1 & \text{if } t_j - t_i \leq \beta_{ij}^h - T \text{ is in effect} \\ 0 & \text{if } t_j - t_i \geq \alpha_{ij}^h - T \text{ is in effect} \\ & \qquad \text{if } \beta_{ij}^h > 0 \text{ for } i = 0, \ldots, n-1; \\ & \qquad\qquad j = i+1, \ldots, n. \end{cases}$$

$$y_{ij} = \begin{cases} 1 & \text{if } t_j - t_i \geq \alpha_{ij}^h + T \text{ is in effect} \\ 0 & \text{if } t_j - t_i \leq \beta_{ij}^h + T \text{ is in effect} \\ & \qquad \text{if } \alpha_{ij}^h < 0 \text{ for } i = 0, \ldots, n-1; \\ & \qquad\qquad j = i+1, \ldots, n. \end{cases}$$

With the above notation, the multi-hoist scheduling problem with time windows can be formulated as the mixed-integer linear programming model below.

$$\text{(MIP)} \qquad \min \ T \tag{7}$$

$$\text{s.t.} \quad a_i + r_{i-1} - (1 - x_{i-1,i})M \leq t_i - t_{i-1} \leq b_i + r_{i-1} + (1 - x_{i-1,i})M, \tag{8}$$

$$a_i + r_{i-1} - x_{i-1,i}M \leq T + t_i - t_{i-1} \leq b_i + r_{i-1} + x_{i-1,i}M, \quad i = 1, \ldots, n. \tag{9}$$

$$t_j - t_i \geq \alpha_{ij}^{q-p} - (3 - z_{ip} - z_{jq} - x_{ij})M, \tag{10}$$

$$t_j - t_i \leq \beta_{ij}^{q-p} + (2 - z_{ip} - z_{jq} + x_{ij})M, \quad i, j = 0, 1, \ldots, n;\ i < j;\ p = l_i, \ldots, u_i;\ q = l_j, \ldots, u_j. \tag{11}$$

$$t_j - t_i \leq T + \beta_{ij}^{q-p} + (3 - z_{ip} - z_{jq} - x_{ij})M, \tag{12}$$

$$t_j - t_i \geq \alpha_{ij}^{q-p} - T - (2 - z_{ip} - z_{jq} + x_{ij})M, \quad i, j = 0, 1, \ldots, n;\ i < j;\ p = l_i, \ldots, u_i;\ q = l_j, \ldots, u_j;\ \beta_{ij}^{q-p} \leq 0 \leq \alpha_{ij}^{q-p}. \tag{13}$$

$$t_j - t_i \leq \beta_{ij}^{q-p} - T + (3 - z_{ip} - z_{jq} - y_{ij})M, \tag{14}$$

$$t_j - t_i \geq \alpha_{ij}^{q-p} - T - (2 - z_{ip} - z_{jq} + y_{ij})M, \quad i, j = 0, 1, \ldots, n;\ i < j;\ p = l_i, \ldots, u_i;\ q = l_j, \ldots, u_j;\ \beta_{ij}^{q-p} > 0. \tag{15}$$

$$t_j - t_i \geq \alpha_{ij}^{q-p} + T - (3 - z_{ip} - z_{jq} - y_{ij})M, \tag{16}$$

$$t_j - t_i \leq T + \beta_{ij}^{q-p} + (2 - z_{ip} - z_{jq} + y_{ij})M, \quad i, j = 0, 1, \ldots, n;\ i < j;\ p = l_i, \ldots, u_i;\ q = l_j, \ldots, u_j;\ \alpha_{ij}^{q-p} < 0. \tag{17}$$

$$\sum_{k=l_i}^{u_i} z_{ik} = 1, \quad i = 0, 1, \ldots, n. \tag{18}$$

$$t_0 = 0. \tag{19}$$

$$0 \leq t_i < T, \quad i = 1, \ldots, n. \tag{20}$$

$$z_{ik} \in \{0, 1\}, \quad i = 0, 1, \ldots, n, k = l_i, l_i + 1, \ldots, u_i. \tag{21}$$

$$x_{ij} \in \{0, 1\}, \quad i = 0, \ldots, n-1;\ j = i+1, \ldots, n. \tag{22}$$

$$y_{ij} \in \{0, 1\}, \quad i = 0, \ldots, n-1;\ j = i+1, \ldots, n. \tag{23}$$

The objective function of this model is simple and the objective is clearly to minimize the cycle length.

Constraints (8) and (9) guarantee the processing time in each tank being within its time limits. If $x_{i-1,i} = 1$, the ith processing stage starts and ends in the same cycle and the corresponding processing time can be expressed by $t_i - (t_{i-1} + r_{i-1})$; Constraint (8) ensures the processing time within the limits while Constraint (9) is redundant due to the large value of M. Otherwise, when $x_{i-1,i} = 0$, the ith processing stage starts in one cycle and ends in the next. Therefore, the corresponding processing time can be expressed as $t_i + T - (t_{i-1} + r_{i-1})$; Constraint (9) ensures the processing time limits while Constraint (8) is redundant.

Constraints (10) to (17) are move-pair constraints to guarantee feasible hoist routes without collision. For a pair of moves i and j performed by hoist p ($z_{ip} = 1$) and q ($z_{jq} = 1$), respectively, Constraints (10) and (11) ensure that move j is not in the prohibited region of move i in the same cycle, while Constraints (12) to (17) ensure that move j is not in the prohibited regions of move i in the previous and next cycles. Depending on the signs of α_{ij}^h and β_{ij}^h, some of these constraints will take effect: Constraints (10) to (13) are for the case with $\beta_{ij}^h \leq 0 \leq \alpha_{ij}^h$; when $\beta_{ij}^h > 0$, Constraints (10), (11), (14), and (15) will be valid to avoid hoist collisions; Constraints (10), (11), (16), and (17) will be valid if $\alpha_{ij}^h < 0$. Constraints (18) ensure that each move is performed by one of the hoists restricted by the individual-move constraints (Constraints (3)). Therefore, Constraints (10) to (18) ensure that all moves in the cycle are performed and that the hoist routes are feasible (no-collisions).

Constraints (19) to (22) are non-negativity and binary constraints.

It can be seen that in this new model, the way of defining most binary variables and the way of formulating many of the constraints are different from those for the previous model in Leung and Zhang (2003). On one hand, this new way of modeling requires more complicated data preparation, especially the calculation of α and β values. On the other hand, with the prepared data the new model itself is much more compact as presented above. In addition, depending on the corresponding α and β values calculated, for any pair of moves, only one among the three constraint groups, (12) and (13), (14) and (15), or (16) and (17), will appear in the model. Thus, the size of the new model is much smaller. For example, the number of binary variables in our module is only 183 and the number of constraints is only 399 for the benchmark problem in Philips and Unger (1976), compared to 508 and 2016, respectively, in the previous model. Furthermore, each constraint in the new model only involves a few variables and so the matrix of the model is sparser. Therefore, the new model can be solved more efficiently, as will be seen from the computational results in Section 6. Moreover, the new model considers safety distance between hoists, uses the more realistic three-segment representation for moves as described in Section 2, and avoids hoist collisions by ensuring that the moves are in

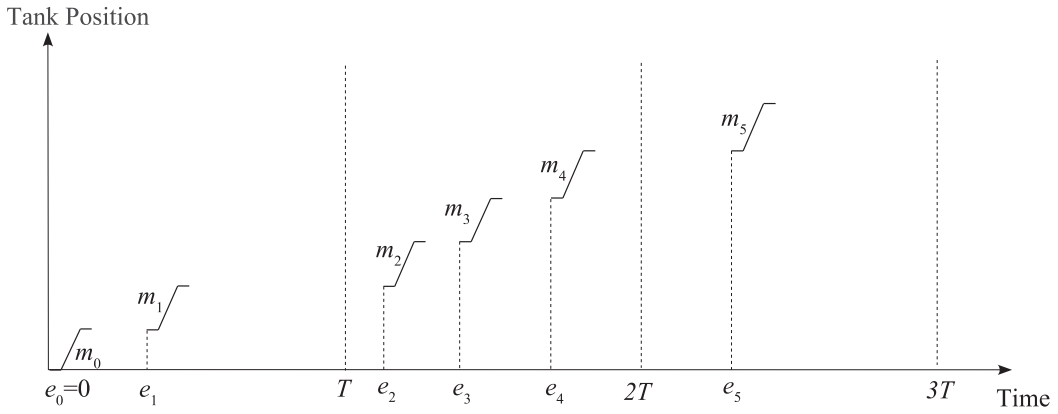

Fig. 8. The relationship between e_i and t_i.

each other's non-prohibited regions rather than by checking their sequencing.

The need for considering collision-avoidance constraints distinguishes the multi-hoist problem from the single-hoist problem. These constraints make the multi-hoist problem difficult to model and solve. Due to different possible relative positions of the moves, these constraints need to be presented in different cases. The three-segment representation of moves increases the number of cases. Our modeling method enables most work for handling the cases to be done in the data processing stage and allows a more efficient branch-and-bound procedure to solve the model itself.

5. A branch-and-bound algorithm

In this section we propose a branch-and-bound solution strategy based on the characteristics of the problem. The key for this solution strategy is our branching scheme on the binary variables $x_{i-1,i}$, $i = 1, 2, \ldots, n$. We start from showing the effect of fixing the values for the $x_{i-1,i}$.

Consider a carrier that begins its m_0 at time 0. We denote e_i as the starting time of the move i corresponding to this carrier. The e_i are different from t_i because the t_i are the starting time of moves in the cycle $[0, T)$, which may be corresponding to different carriers. If the processing time for the kth stage is τ_k, e_i can be calculated easily using the equation

$$e_i = \sum_{k=1}^{i}(r_{k-1} + \tau_k) \quad i = 1, 2, \ldots, n \text{ and } e_0 \equiv 0. \quad (24)$$

Considering the limits for the processing times, we have the following inequality:

$$\sum_{k=1}^{i}(r_{k-1} + a_k) \le e_i \le \sum_{k=0}^{i}(r_{k-1} + b_k), \quad i = 1, 2, \ldots, n \text{ and } e_0 \equiv 0. \quad (25)$$

The relationship between e_i and t_i can be expressed as

$$t_i = e_i \bmod T \text{ and } e_i = t_i + \theta_i T \text{ where } \theta_i = \lfloor e_i / T \rfloor, \quad i = 0, 1, \ldots, n. \quad (26)$$

Figure 8 shows an example system with five chemical tanks. For a carrier that starts its m_0 at time 0 in the example, the moves 0 and 1 start in the cycle $[0, T)$ and we have $t_0 = e_0$ and $t_1 = e_1$, moves 2, 3, and 4 start in the cycle $[T, 2T)$ and we have $t_k = e_k - T$ for $k = 2, 3, 4$. The last move starts in the third cycle $[2T, 3T)$ and we have $t_5 = e_5 - 2T$.

Based on the above analysis and the definition of $x_{i-1,i}$, it can be concluded that for a pair of consecutive moves $i - 1$ and i of the same carrier, the corresponding e_{i-1} and e_i are in the same cycle if $x_{i-1,i} = 1$; otherwise, if $x_{i-1,i} = 0$, e_{i-1} is in one cycle and e_i is in the next. In addition, we can conclude that x_{jk} is equal to one for any pair of moves j and k ($j < k$) with e_j and e_k in the same cycle. For the example shown in Fig. 8, we have $x_{1,2} = x_{4,5} = 0$ and $x_{0,1} = x_{2,3} = x_{3,4} = 1$. We can also obtain $x_{2,4} = 1$ for this example. Therefore, if we fix $x_{i-1,i}$, whether the e_i of a pair of moves are in the same cycle becomes known.

5.1. *The branching scheme*

To solve the problem with the branch-and-bound method, we propose to branch on the binary variables $x_{i-1,i}$, $i = 1, 1, \ldots, n$. The search tree starts from the root node that stands for the original problem and defines that move 0 of each carrier starts a new cycle. The root node is in layer 0 in the searching tree and labeled as node 0.

All the nodes branched out directly from the root node are in layer 1. These nodes are labeled as $1, 2, \ldots, n + 1$. The node labeled as i represents the subproblem with $x_{i-1,i} = 0$ and $x_{j-1,j} = 1$ for $0 < j < i$; i.e., all of the e_j ($0 < j < i$) are in the same cycle as e_0 but e_i is in the next cycle. Note that $i = 1$ is a special case where no j would satisfy the condition $0 < j < i$. Corresponding to this case, the node labeled as 1 represents the subproblem with $x_{0,1} = 0$; i.e., e_1 is in the cycle next to the one containing e_0. It is also

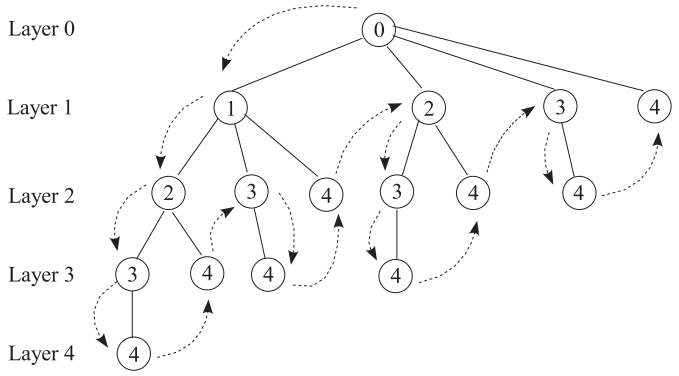

Fig. 9. Complete search tree for a problem with $n = 3$.

worthwhile to note the special child node labeled as $n + 1$, which means all of the e_j up to e_n are in the same cycle and $x_{j-1,j} = 1$ for all $j \le n$. This is a leaf node and needs not be further branched. Corresponding to such a node, we define $e_{n+1} \equiv \infty$ and $x_{n,n+1} \equiv 0$ for presentation convenience.

Each layer-1 node, except the leaf node, can be branched to generate its own child nodes. All of the child nodes of layer-1 nodes are in layer 2. In general, all child nodes of the nodes in layer $(k-1)$ constitute layer k, $k = 1, \ldots, n+1$. Branching a node labeled as h in layer $(k-1)$ generates $(n-h+1)$ child nodes in layer k and these child nodes are labeled as $h+1, \ldots, n$, and $n+1$. A node labeled as i in layer k branched from a node labeled as h in layer $(k-1)$ represents the subproblem with $x_{i-1,i} = 0$ and $x_{j-1,j} = 1$ for $h < j < i$; i.e., the corresponding e_i is in a new cycle $[kT, (k+1)T)$ but all e_j, $h \le j < i$, are in cycle $[(k-1)T, kT)$ and, consequently, $e_h < e_{h+1} < \cdots < e_{i-1} < kT \le e_i$.

The complete searching tree has $n + 2$ layers. Only those nodes labeled as $n + 1$ are leaf nodes. We use a depth-first searching strategy for our branch-and-bound algorithm. A complete search tree for an example system consisting of three chemical tanks is given in Fig. 9. The arrowed dashed lines in Fig. 9 indicate the searching sequence in the branch-and-bound algorithm.

5.2. *Simple bounds*

For the root node, we define the initial lower bound as $\max_{i=1,2,\ldots,n}\{a_i\}$ because we must guarantee that the throughput should not exceed the capacity of any chemical tank in the system. The upper bound is set to be $e_n + r_n + |w_{s_{n+1}}|/\lambda$, which is the cycle length corresponding to the simple feasible schedule in which a carrier enters the system after the previous one leaves.

Now consider a node labeled as γ_π in layer π of the search tree with its preceding nodes in layer j ($j = 0, 1, \ldots, \pi - 1$) labeled as γ_j. For each node γ_j, we have $e_{\gamma_j - 1} < jT \le e_{\gamma_j}$, where we define $e_{-1} \equiv 0$ for mathematical consistency. Therefore, as shown in Fig. 10, there are at least $\pi - j$ cycle lengths between $e_{\gamma_j - 1}$ and e_{γ_π}. In addition, the time between e_{γ_j} and $e_{\gamma_\pi - 1}$ is no more than $\pi - j$ cycle lengths. This analysis leads to the following inequalities:

$$e_{\gamma_\pi - 1} - e_{\gamma_j} \le (\pi - j)T \le e_{\gamma_\pi} - e_{\gamma_j - 1}, \quad j = 0, 1, \ldots, \pi - 1. \tag{27}$$

Considering Equation (24) and the limits for the processing times, we can obtain the following simple bounds associated with node γ_π:

$$T \ge \max\left\{\left(\sum_{i=\gamma_j+1}^{\gamma_\pi - 1}(r_{i-1} + a_i)\right)\Big/(\pi - j) \,|\, j = 0, \ldots, \pi - 1\right\}, \tag{28}$$

$$T \le \min\left\{\left(\sum_{i=\gamma_j}^{\gamma_\pi}(r_{i-1} + b_i)\right)\Big/(\pi - j) \,|\, j = 0, \ldots, \pi - 1\right\}. \tag{29}$$

5.3. *An improved lower bound*

The lower bound in Equation (28) is simple and easy to calculate. When a new node is generated in the branch-and-bound process, we first try to fathom it based on the simple bounds calculated by Equations (28) and (29). The upper bound is also updated whenever a feasible solution is found. If a node cannot be fathomed using the simple bounds, a tighter lower bound can be calculated by solving a much reduced mixed-integer programming model associated with this node.

The model to obtain the lower bound for a node can be derived by modifying model (MIP). Consider a node labeled as γ_π (in layer π), whose preceding node in layer j is node γ_j, where $j = 0, 1, \ldots, \pi - 1$. To obtain a lower bound for the node, we remove all constraints involving moves after move γ_π from model (MIP) and consider only the constraints among moves $1, 2, \ldots, \gamma_\pi$. In addition, some x_{ij} values are fixed for the node as discussed in the previous subsection and therefore can be removed from the model. As a result, the lower bound model for the node is much smaller than (MIP). The following outlines

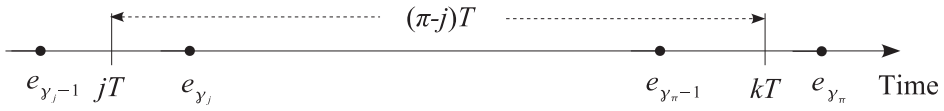

Fig. 10. The relationship between e_{γ_j} and e_{γ_π}.

the process of reducing (MIP) to obtain the lower bound model for node γ_π.

1. Replace the parameter n by $\min\{n, \gamma_\pi\}$.
2. Determine the values for some x_{ij}:

$$x_{i-1,i} = 0 \quad \text{for } i \in \{\gamma_1, \ldots, \gamma_\pi\}.$$
$$x_{i,j} = 1 \quad \text{for } (i,j) \in \bigcup_{k=1,\ldots,\pi} \{(i,j) | \gamma_{k-1} \le i < j \le \gamma_k - 1\}.$$

3. Remove the above x_{ij} variables from the model by substituting them with their fixed values.

Note that fixing the values for the x_{ij} variables enables some constraints to be removed: those Constraints (8), (10), and (12) corresponding to x_{ij} variables with a fixed value of zero become redundant and can be removed; those Constraints (9), (11), and (13) corresponding to x_{ij} variables with a fixed value of one become redundant and can be removed; those Constraints (22) for x_{ij} variables with fixed values can be removed.

The new model associated with node γ_π just considers a partial system involving the first γ_π processing stages. The model may be infeasible due to fixing the $x_{i-1,i}$ variables. The node will be fathomed if the corresponding model is infeasible. Otherwise, we get a lower bound for the node. The feasible optimal cycle length for the model corresponding to a leaf node in the search tree is a global upper bound because the model is for the whole system and the solution is feasible. Note that the upper bound calculated by Inequality (29) is just an upper bound for the corresponding node rather than a global upper bound for all nodes.

6. Computation results and discussion

To test the efficiency of the model and the proposed branch-and-bound algorithm, we apply them to a well-known benchmark problem and a set of randomly generated problems. All computations in this section are performed on a PC with a Pentium Core2Duo 2.0GHz CPU and 2G memory. The mixed-integer programming models are solved using the software package CPLEX 11.11.

We first present the computation results for a benchmark problem in Phillips and Unger (1976). The problem is a real industrial application with data as shown in Table 1. The tank position data are derived from the original travel data in Phillips and Unger (1976).

Table 2. The optimal cycle lengths for the benchmark problem

	Number of hoists			
	1	*2*	*3*	*4*
Optimal cycle length	514	242	216	216

We solve the problems with one, two, three, and four hoists. Table 2 shows the optimal cycle lengths for problems with different numbers of hoists. Please note that the tanks for stages 2 and 4 are at the same position, indicating that they are the same multi-function tank. As in most previous multi-hoist studies, here we have considered these two stages as being in separate tanks very close to each other. That is why the optimal cycle length with a single hoist is 514 rather than 521 as it would be if the multi-funtion tank constraint was added. Constraints for handling multi-function tanks can be found in Liu *et al.* (2002).

When two hoists are used, the optimal cycle length decreases to about 47% of that in the one-hoist case. When the third hoist is added, the optimal cycle length decreases further to around 42% of that in the one-hoist case. The decrease in the cycle length by adding the third hoist is less than the decease when adding the second hoist. The optimal cycle length remains unchanged when the fourth hoist is added. This is in line with the intuition that the optimal cycle length decreases with increased handling resources until reaching a level where the handling resources are sufficient and the processing resources become bottleneck. Adding further hoists may actually increase the cycle length due to more interferences among them. Figure 11 shows the time-way diagram of an optimal hoist schedule for the benchmark problem with three hoists.

Table 3 shows the CPU times for solving the benchmark problems by solving the (MIP) model directly and using the proposed branch-and-bound algorithm with and without the improved lower bound. We also implemented and ran the model presented in Leung and Zhang (2003). The CPU times are listed in Table 3 for comparison. The results show that the new model is much more efficient than the Leung–Zhang model especially when there are multiple hoists. The proposed branch-and-bound algorithm solves

Table 1. Data for the benchmark problem ($n = 12$, $\lambda = 1$, $v = 1$, $\mu_i = 8.5$, $\eta_i = 11.5$, $d = 1$, $w_l = 0$, $w_r = 29$)

	i													
	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*
w_i	0	6	8	10	11	14	14	16	19	22	24	26	29	
s_i	0	4	5	7	6	8	9	10	11	12	1	2	3	0
a_i	120	150	90	120	90	30	60	60	45	130	120	90	30	
b_i	∞	200	120	180	125	40	120	120	75	∞	∞	120	60	

Table 3. The CPU times (in seconds) used for the benchmark problem

	Number of hoists			
	1	*2*	*3*	*4*
Proposed model (MIP)	0.31	23.87	4.42	0.64
Leung–Zhang model	1.6	1565	4507	47.3
B&B without improved lower bound	0.85	4.85	4.31	6.74
B&B with improved lower bound	0.34	2.67	0.32	1.91

B&B = branch-and-bound.

Table 4. The average optimal cycle length for random problems

	Number of hoists								
	12 Tanks			*16 Tanks*			*20 Tanks*		
	1	*2*	*3*	*1*	*2*	*3*	*1*	*2*	*3*
5–20%	675	303	277	1088	371	281	1517	473	298
20–50%	480	277	271	656	290	272	868	335	272

the problems more efficiently than solving the model directly using the commercial package. The improved lower bound shortens the computation time further.

To test the computation times for different problems, we randomly generated problems with three sizes: 12, 16, and 20 tanks. The lower limit of each processing time, a_i, was randomly generated from the uniform distribution [30, 250]. The b_i values were generated in two different ways to represent different levels of time window restrictions: each b_i was set 5–20% and 20–50% uniformly above its corresponding a_i. The combination of problem sizes (number of tanks) and time windows gave six different problem sets, each with 20 problem instances generated. Each instance was solved for the one, two, and three hoist cases.

In each problem, the tanks were numbered sequentially from tank 0, and their positions were $w_i = i$, $i = 0, 1, \ldots, n$, and $w_{n+1} = 0$. The hoist parameters were $d = 1.5$, $\nu = 0.5$, $\lambda = 0.8$, and $\mu_i = \eta_i = 10$. The left-most and right-most positions of the track were $w_l = 0$ and $w_r = n$. The carrier route was generated as follows: n binary numbers were randomly generated for each tank in the line; then carriers visit the tanks with 1 from left to right and then the tanks with 0 from right to left. For example, if $n = 8$ and the generated numbers were 01101001, the carriers went from tank 0 (loading/unloading station) to tank 2, then 3, 5, 8, 7, 6, 4, 1, and then back to tank 0. This way of generating carrier routes is in line with

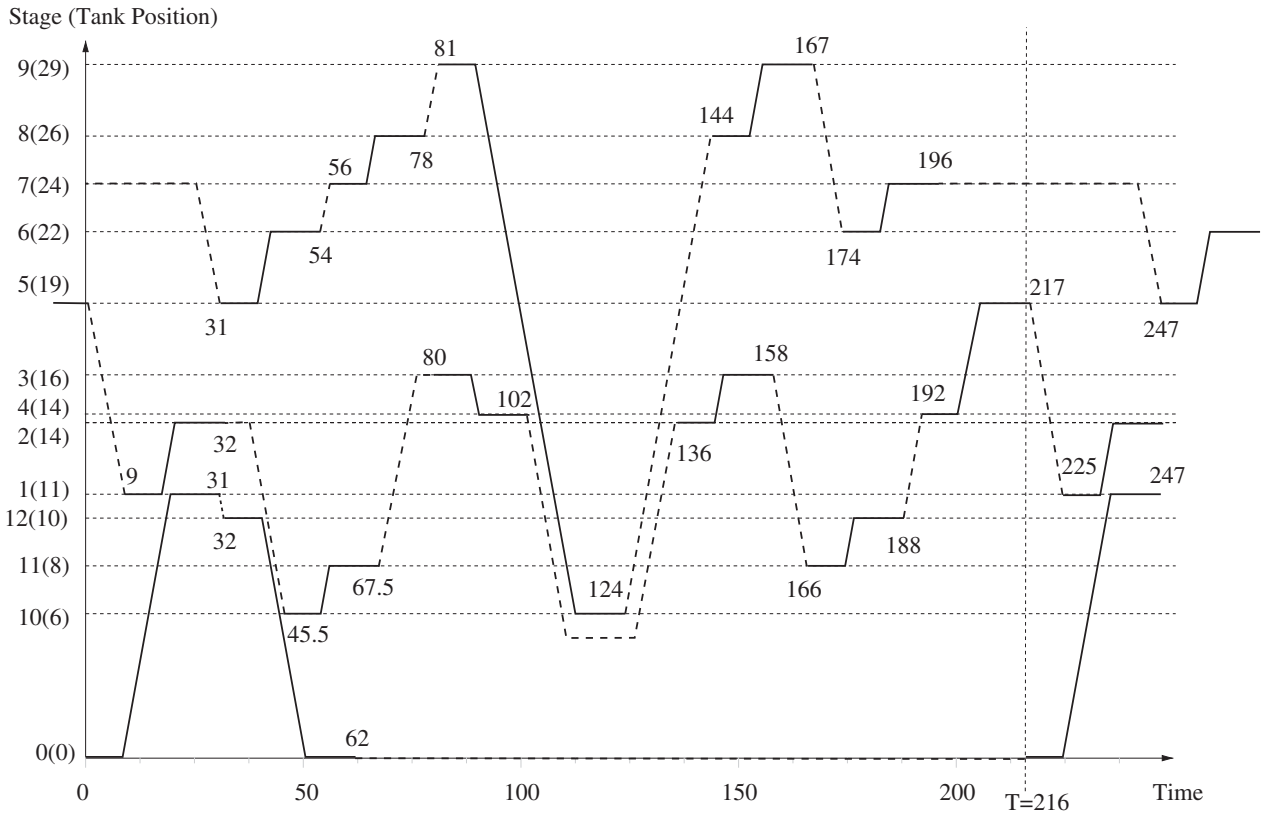

Fig. 11. An optimal solution to the benchmark problem with three hoists.

Table 5. The average CPU time ratios of our model to the Leung–Zhang model for the 12 tank problem

	Number of hoists		
	1	*2*	*3*
5–20%	1/5.5	1/213	1/433
20–50%	1/4.4	1/388	1/131

considerations in design of the systems where the tanks are arranged so that the carriers travel a minimum distance while visiting all of the tanks for processing. Nevertheless, carrier routes generated this way allow overlapping of the moves vertically in the time-way diagram and hence require the collision-avoidance constraints.

Table 4 shows the average optimal cycle length for the randomly generated problems. We can see that the optimal cycle length is sensitive to the processing time windows and decreases when the time windows increase. For the system with smaller time windows, more hoists are needed to avoid the hoists becoming a bottleneck. If there are enough hoists in a system, the optimal cycle length will be close to the lower bound calculated from the maximum lower limit of processing times, $\max\{a_i\} + \mu + d/v + \eta$. At this point, the tanks become a bottleneck and the optimal cycle length remains unchanged or decreases very little even if more hoists are added. This can be observed from the cases with 12 tanks and two or three hoists for 20–50% time windows. The average optimal cycle length decreases by around 2% when the third hoist is added.

Table 5 compares the computation times to solve our model and the Leung–Zhang model using CPLEX. Each cell in the table is the average ratio of the CPU time spent by our model to that for the Leung–Zhang model. Only the problems with 12 tanks are used in the comparison because larger problems cannot be solved by the Leung–Zhang model in a reasonable time. Similar to the result for the benchmark problem, when there is one hoist and no hoist collision constraints, the proposed model can be solved around five times faster than the Leung–Zhang model. When there are multiple hoists, the proposed model solves the problems hundreds of times faster than the Leung–Zhang model, which is very significant. Generally speaking, the new model outperforms the Leung–Zhang model and the time savings is more significant for multi-hoist problems.

Table 6 shows the average CPU times for solving each group of problems by solving the model directly with CPLEX and by using the proposed branch-and-bound algorithm with the improved lower bound. When solving the model with CPLEX, we set CPLEX to branch the decision variables $x_{i-1,i}$ first, which is more efficient and consistent with the branching scheme used in the proposed branch-and-bound method. The symbol "—" in a cell means that some problems in the corresponding group could not be solved to optimum because of insufficient memory. It can be seen that the CPU time is sensitive to the number of tanks, the number of hoists, and processing time window. The computation times increase when the number of tanks or the number of hoists increases. When the processing time window is enlarged, more computation time is needed. Comparing our proposed branch-and-bound algorithm with the general branch-and-bound procedure in CPLEX with similar settings for the branching scheme, the proposed branch-and-bound algorithm outperforms the general package in all cases. The difference becomes larger when the problem size increases.

A number of factors may have contributed to the higher efficiency of the proposed branch-and-bound method. The special branching scheme presented in Section 5 takes advantage of the relationships among the variables. This makes the search tree smaller than that of a standard search tree with $n!$ leaf nodes, as can be seen from Fig. 9. The branching scheme is implemented using a general programming language. The simple bounds are based on processing time constraints and, though not very tight, do not involve solution of mathematical programming models. Therefore, if only the simple bounds are used, the proposed branch-and-bound method does not rely on any solver. After the improved lower bound is used, a reduced (MIP) model is solved at each node which is not fathomed by the simple bound. The reduced (MIP) at a node corresponds to only part of the system and has many variables' values fixed. This is different from a standard linear programming relaxation. The special way of branching and the depth first

Table 6. The average CPU times (in seconds) for random problems

		Number of hoists								
		12 Tanks			*16 Tanks*			*20 Tanks*		
		1	*2*	*3*	*1*	*2*	*3*	*1*	*2*	*3*
5–20%	B&B	0.064	0.25	0.31	0.21	2.56	3.48	0.45	13.73	83
	CPLEX	0.096	0.93	0.76	0.25	45	62	0.52	2332	—
20–50%	B&B	0.31	0.46	0.59	1.45	19.3	11.26	3.16	1896	3084
	CPLEX	0.48	0.93	1.12	3.3	211	443	10	—	—

search strategy also mean that the nodes searched in early part in the solution process tend to correspond to small cycle lengths. Therefore, many of them are likely to be easily verified as being infeasible. Once a feasible solution is found, it tends to be close to optimal and hence provides a good upper bound that in turn helps to fathom later nodes. While we may try to set a standard branch and bound for (MIP) in similar ways, such as prioritizing the variables to branch as we did in the experiment, it is difficult or impossible to implement all of the above features in a standard solver package. This may explain why the proposed branch-and-bound algorithm is more efficient than a standard solver package.

7. Conclusions

In this article, we studied the general multi-hoist cyclic scheduling problem in electroplating lines where the processing times are flexible within given limits. We first analyzed the no-collision requirements between hoists and transformed them to linear constraints between each pair of moves in the cycle $[0, T)$. Incorporating these constraints, the problem was formulated as a new mixed-integer linear programming model. A branch-and-bound algorithm that took advantage of the problem structure was then developed to solve the problem more efficiently. Computational experiments showed that the algorithm can solve practical sized problems in a relatively short time. No-collision requirements for material handling equipment exist in many industrial applications. The approach here to dealing with such requirements can be useful for other problems with no-collision requirements.

Acknowledgement

The research was supported in part by the National Natural Science Foundation of China (grant 70728001) through the Collaborative Research Fund for Overseas Young Scholars.

References

Armstrong, R., Gu, S. and Lei, L. (1996) A greedy algorithm to determine the number of transporters in a cyclic electroplating process. *IIE Transactions*, **28**, 347–355.

Che, A. and Chu, C. (2004) Single-track multi-hoist scheduling problem: a collision-free resolution based on a branch-and-bound approach. *International Journal of Production Research*, **42**, 2435–2456.

Che, A. and Chu, C. (2005) Multi-degree cyclic scheduling of two robots in a no-wait flowshop. *IEEE Transactions on Automation Science and Engineering*, **2**, 173–183.

Crama, Y., Kats, V., Van de Klundert, J. and Levner, E. (2000) Cyclic scheduling in robotic flowshop. *Annals of Operations Research*, **96**, 97–124.

Crama, Y. and Van de Klundert, J. (1997) Robotic flowshop scheduling is strongly NP-complete, in *Ten Years LNMB*, Haneveld, W.K., Vrieze, O.J., and Kallenberg, L.C.M. (eds), CWI Tract, Amsterdam, The Netherlands, pp. 277–286.

Heinrichs, U. and Moll, C. (1997) On the scheduling of one dimensional transport systems. Technical Report 97-277, University of Cologne, Cologne, Germany.

Jiang, Y. and Liu, J. (2007) Multihoist cyclic scheduling with fixed processing and transfer times. *IEEE Transactions on Automation Science and Engineering*, **4**, 435–450.

Karzanov, A.V. and Livshits, E.M. (1978) Minimal quantity of operators for serving a homogeneous linear technological process. *Automation and Remote Control*, **39**, 445–450.

Kats, V. and Levner, E. (1997a) Minimizing the number of robots to meet a given cyclic schedule. *Annals of Operations Research*, **69**, 209–226.

Kats, V. and Levner, E. (1997b) A strongly polynomial algorithm for no-wait cyclic robotic flowshop scheduling. *Operations Research Letters*, **21**, 171–179.

Lei, L., Armstrong, R. and Gu, S. (1993) Minimizing the fleet size with dependent time-window and single-track constraints. *Operations Research Letters*, **14**, 91–98.

Lei, L. and Wang, T.J. (1989) A proof: the cyclic hoist scheduling problem is NP-complete. Working Paper 89-0016, Rutgers University, Piscataway, NJ.

Lei, L. and Wang, T.J. (1991) The minimum common-cycle algorithm for cyclic scheduling of two material handling hoists with time window constraints. *Management Science*, **37**, 1629–1639.

Leung, J. and Levner, E. (2006) An efficient algorithm for multi-hoist cyclic scheduling with fixed processing times. *Operations Research Letters*, **34**, 465-472.

Leung, J. and Zhang, G. (2003) Optimal cyclic scheduling for printed circuit board production lines with multiple hoists and general processing sequence. *IEEE Transactions on Robotics and Automation*, **19**, 480–484.

Leung, J., Zhang, G., Yang, X., Mak, R. and Lam, K. (2004) Optimal cyclic multi-hoist scheduling: a mixed integer programming approach. *Operations Research*, **52**, 965–976.

Liu, J. and Jiang, Y. (2005) An efficient optimal solution to the two-hoist no-wait cyclic scheduling problem. *Operations Research*, **53**, 313-327.

Liu, J., Jiang, Y. and Zhou, Z. (2002) Cyclic scheduling of a single hoist in extended electroplating lines: a comprehensive integer programming solution. *IIE Transactions*, **34**, 905–914.

Manier, M. and Bloch, C. (2003) A classification for hoist scheduling problems. *The International Journal of Flexible Manufacturing Systems*, **15**, 37–55.

Phillips, L.W. and Unger, P.S. (1976) Mathematical programming solution of a hoist scheduling program. *AIIE Transactions*, **28**, 219–225.

Serafini, P. and Ukovich, W. (1989) A mathematical model for periodic scheduling problems. *SIAM Journal of Discrete Mathematics*, **2**, 550–581.

Zhou, Z. and Liu, J. (2008) A heuristic algorithm for the two-hoist cyclic scheduling problem with overlapping hoist coverage ranges. *IIE Transactions*, **40**, 782–794.

Appendix: Formulae of α_{ij}^h and β_{ij}^h in different cases

For $h = 0$, the two moves are performed by the same hoist and there must be enough time for the empty hoist to travel between the two moves. α_{ij}^0 equals the starting time of m_j that touches the prohibited boundary after m_i minus the starting time of m_i. From Fig. 4, it is clear that α_{ij}^0 can be

calculated as follows:

$$\alpha_{ij}^0 = r_i + \left(w_{s_j} - w_{s_{i+1}}\right)/\lambda, \text{ in case of } w_{s_j} \geq w_{s_{i+1}},$$
$$\alpha_{ij}^0 = r_i + \left(w_{s_{i+1}} - w_{s_j}\right)/\lambda, \text{ in case of } w_{s_j} < w_{s_{i+1}}.$$

In both cases, α_{ij}^0 can be expressed as

$$\alpha_{ij}^0 = r_i + \left|w_{s_j} - w_{s_{i+1}}\right|/\lambda \tag{A1}$$

β_{ij}^0 equals the starting time of the m_j that touches the prohibited boundary before m_i minus the starting time of m_i. From Fig. 4, it can be seen that β_{ij}^0 can be calculated as $\beta_{ij}^0 = -r_j - |w_{s_{j+1}} - w_{s_i}|/\lambda$.

Comparing the formulae of α and β, we can obtain the following relationship: $\beta_{ij}^0 = -\alpha_{ji}^{-0}$.

For $h > 0$, we need to consider all possible cases of relative tank positions involved in a move pair. For move i, its start tank position can be either lower or higher than its

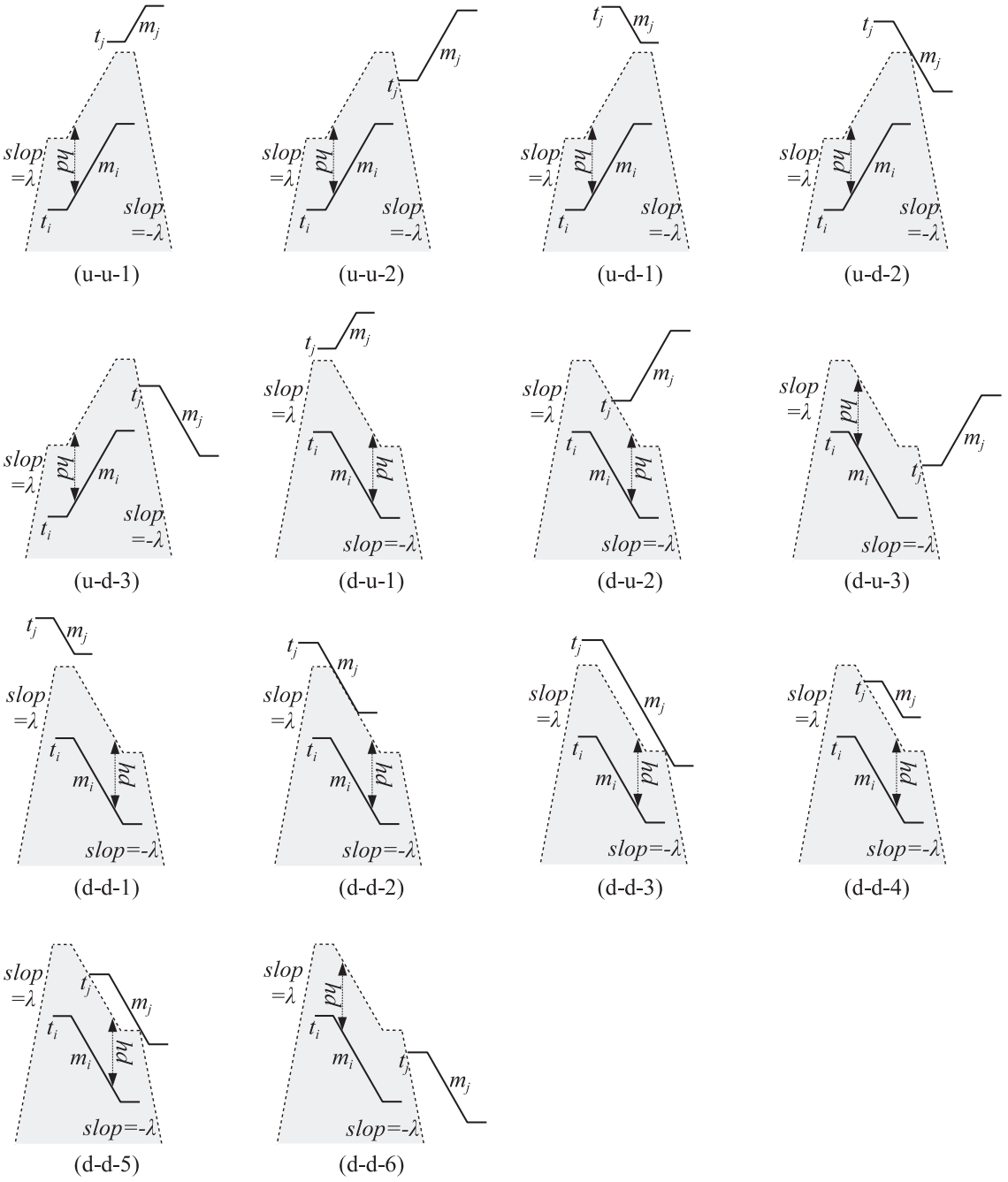

Fig. A1. Different cases of relative tank positions involved in a move pair for $h > 0$.

end tank position; i.e., move i can appear either going up or going down in the time-way diagram. Similarly, move j can go either up or down in the time-way diagram. The different moving directions of the two moves give four possible combinations: up-up, up-down, down-up, and down-down. For each combination, there are also only a limited relative relations between the tank positions of move i and those of move j. Considering all of these, we can list all possible cases about the relations of the move pair. For calculating α_{ij}^h, we can look at the possible cases in which move j touches the right border of the prohibited region of move i. We briefly explain below all of the different cases illustrated in Fig. A1.

For combination up-up, there are only two possible cases. If the start point of move j is above the end point of i plus hd (i.e., above $w_{s_{i+1}} + hd$), then move j will never touch the prohibited region of i, as shown in case (u-u-1) in Fig. A1. Otherwise, the start point of j touches the right border line of the prohibited region, as shown in case (u-u-2).

For combination up-down: If the start point of move j is above $w_{s_{i+1}} + hd$, there may only be two cases depending on whether the end point of j is below $w_{s_{i+1}} + hd$ or not. These are shown as cases (u-d-1) and (u-d-2). If the start point of move j is below $w_{s_{i+1}} + hd$, the start point of j touches the right border line of the prohibited region, as shown in case (u-d-3).

For combination down-up: If the start point of move j is above the start point of i plus hd (i.e., above $w_{s_i} + hd$), then move j will never touch the prohibited regoion of i, as shown in case (d-u-1). Otherwise, the start point of j will touch one of the two segments on the right border of the prohibited region, depending on whether the start point of move j is below $w_{s_{i+1}} + hd$ or not, as shown in cases (d-u-2) and (d-u-3).

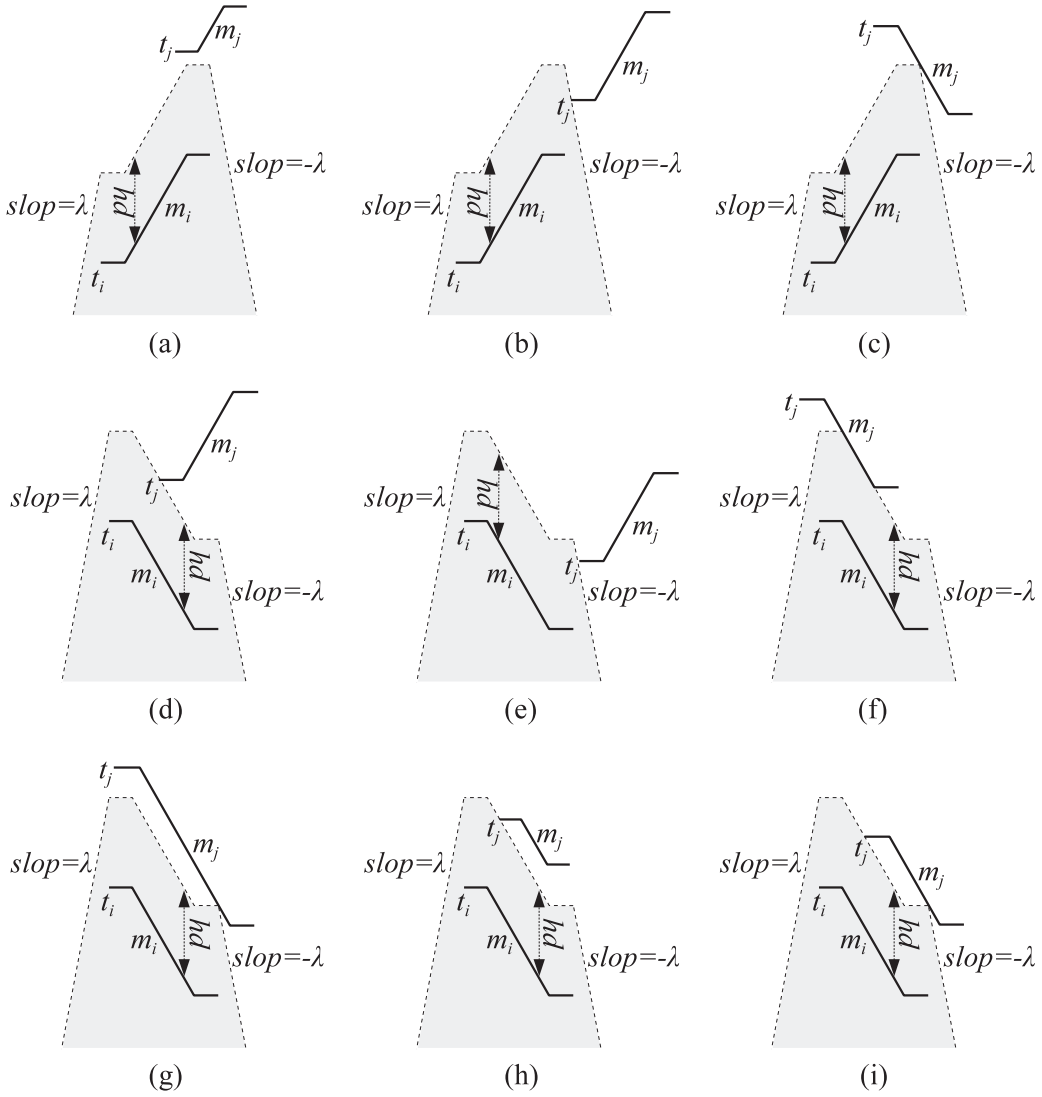

Fig. A2. Consolidated cases of relative tank positions involved in a move pair for $h > 0$.

For combination down-down, there are more cases. If the start point of move j is above $w_{s_i} + hd$, depending on the position of its end point, either the move never touches the prohibited regoion of i (case d-d-1) or it touches the prohibited region on the right in different ways (cases d-d-2 and d-d-3). If the start point of move j is between $w_{s_i} + hd$ and $w_{s_{i+1}} + hd$, there may also be two cases (d-d-4 and d-d-5) depending on the position of its end point. If the start point of move j is below $w_{s_{i+1}} + hd$, its start point touches the lower segement of the right border of the prohibited region.

In summary, Fig. A1 includes all different cases for calculating α_{ij}^h. Note that some of these cases can be combined and viewed as the same case. For example, in cases (u-u-1), (u-d-1), (d-u-1), and (d-d-1), move j never touches the prohibited region of i. Therefore, they can be combined and the combined cases can be expressed using the condition $\max\{w_{s_i}, w_{s_{i+1}}\} + hd \le \min\{w_{s_j}, w_{s_{j+1}}\}$. Similarly, cases (u-u-2) and (u-d-3) can be combined as one. Figure A2 illustrates all distinctive cases after the consolidation. The relations between the original cases in Fig. A1 and the consolidated cases in Fig. A2 are as follows.

Cases (u-u-1), (u-d-1), (d-u-1), and (d-d-1) are combined and represented by case (a); cases (u-u-2), and (u-d-3) are combined and represented by case (b); case (u-d-2) is the same as case (c); case (d-u-2) is the same as case (d); cases (d-u-3) and (d-d-6) are combined and represented by case (e); cases (d-d-2), (d-d-3), (d-d-4), and (d-d-5) are the same as cases (f), (g), (h), and (i), respectively. Though we may further combine case (b) with case (e) and case (d) with case (h), we keep them as separate cases here for clarity.

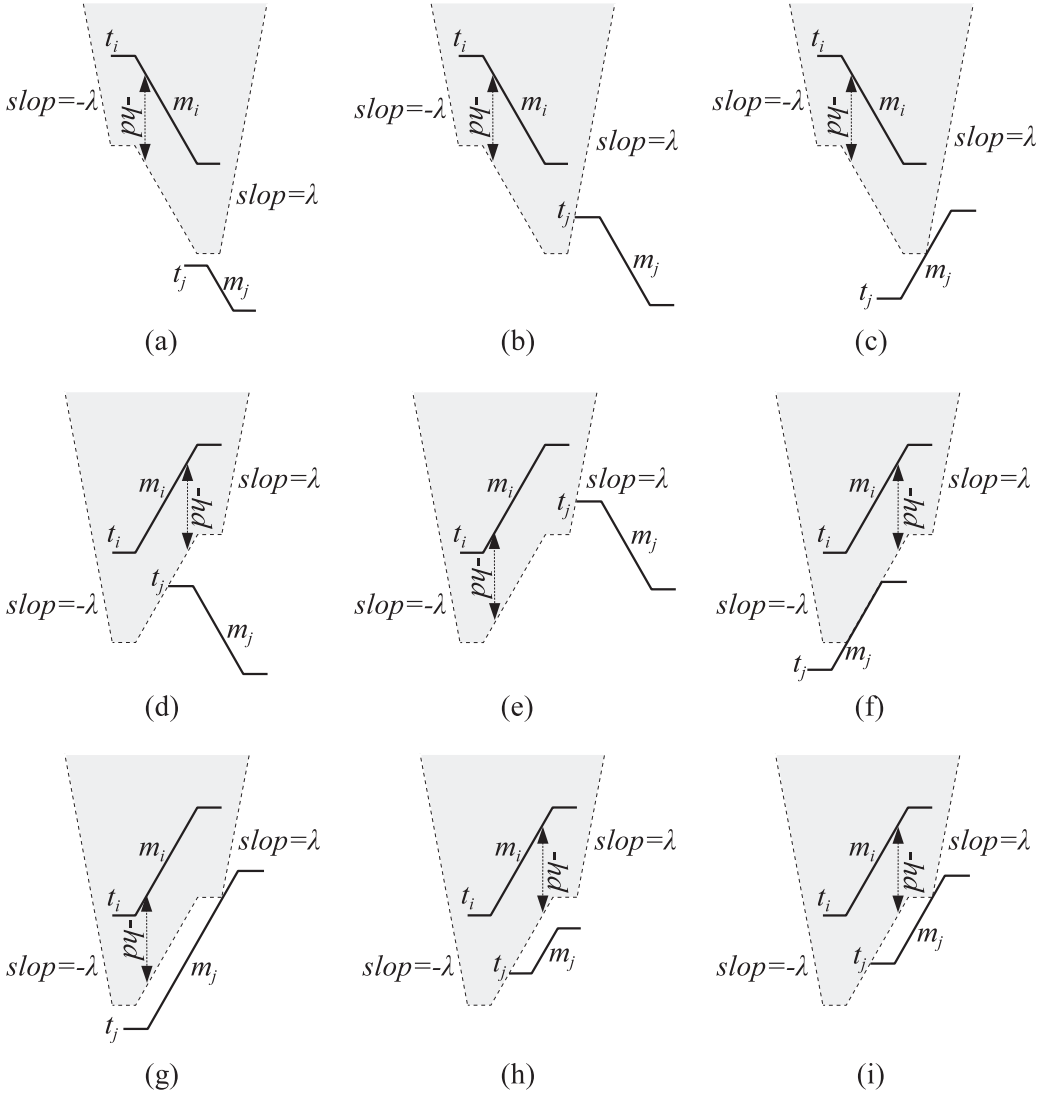

Fig. A3. Consolidated cases of relative tank positions involved in a move pair for $h < 0$.

The conditions for the consolidated cases in Fig. A2 are listed below.

(a) $\max\{w_{s_i}, w_{s_{i+1}}\} + hd \le \min\{w_{s_j}, w_{s_{j+1}}\}$.
(b) $w_{s_i} < w_{s_{i+1}}, w_{s_j} < w_{s_{i+1}} + hd$.
(c) $w_{s_i} < w_{s_{i+1}}, w_{s_j} > w_{s_{j+1}}, w_{s_{j+1}} < w_{s_{i+1}} + hd \le w_{s_j}$.
(d) $w_{s_i} > w_{s_{i+1}}, w_{s_j} < w_{s_{j+1}}, w_{s_{i+1}} + hd \le w_{s_j} < w_{s_i} + hd$.
(e) $w_{s_i} > w_{s_{i+1}}, w_{s_j} < w_{s_{i+1}} + hd$.
(f) $w_{s_i} > w_{s_{i+1}}, w_{s_j} > w_{s_{j+1}}, w_{s_{i+1}} + hd \le w_{s_{j+1}} < w_{s_i} + hd \le w_{s_j}$.
(g) $w_{s_j} \ge w_{s_i} + hd > w_{s_{i+1}} + hd > w_{s_{j+1}}$.
(h) $w_{s_i} + hd > w_{s_j} > w_{s_{j+1}} \ge w_{s_{i+1}} + hd$.
(i) $w_{s_i} + hd > w_{s_j} \ge w_{s_{i+1}} + hd > w_{s_{j+1}}$.

From Fig. A2, we can get the following formula for determining α_{ij}^h in different cases with $h > 0$:

$$\alpha_{ij}^h = \begin{cases} -\infty & \text{for case (a)} \\ r_i + \left(w_{s_{i+1}} + hd - w_{s_j}\right)/\lambda & \text{for cases (b), (e)} \\ r_i - \mu_j - \left[w_{s_j} - (w_{s_{i+1}} + hd)\right]/\nu & \text{for cases (c), (g)} \\ \mu_i + \left(w_{s_i} + hd - w_{s_j}\right)/\nu & \text{for cases (d), (h)} \\ \mu_i - \left[w_{s_j} - (w_{s_i} + hd)\right]/\nu - \mu_j & \text{for case (f)} \\ \mu_i + \left(w_{s_i} + hd - w_{s_j}\right)/\nu & \text{for case (i)} \\ \quad + \max\{0, \eta_i - \mu_i\} & \end{cases} \tag{A2}$$

For case (a), there is no restriction between t_i and t_j. We assign $-\infty$ to α_{ij}^h for the sake of generality.

For $h < 0$, we can list all different cases and combine them in the similar way as above. All of the consolidated cases are shown in Fig. A3, and the conditions for them are listed below.

(a) $\min\{w_{s_i}, w_{s_{i+1}}\} + hd \ge \max\{w_{s_j}, w_{s_{j+1}}\}$.
(b) $w_{s_i} > w_{s_{i+1}}, w_{s_j} > w_{s_{i+1}} + hd$.
(c) $w_{s_i} > w_{s_{i+1}}, w_{s_j} < w_{s_{j+1}}, w_{s_j} \le w_{s_{i+1}} + hd < w_{s_{j+1}}$.
(d) $w_{s_i} < w_{s_{i+1}}, w_{s_j} > w_{s_{j+1}}, w_{s_i} + hd < w_{s_j} \le w_{s_{i+1}} + hd$.
(e) $w_{s_i} < w_{s_{i+1}}, w_{s_j} > w_{s_{i+1}} + hd$.
(f) $w_{s_i} < w_{s_{i+1}}, w_{s_j} < w_{s_{j+1}}, w_{s_j} \le w_{s_i} + hd < w_{s_{j+1}} \le w_{s_{i+1}} + hd$.
(g) $w_{s_j} \le w_{s_i} + hd < w_{s_{i+1}} + hd < w_{s_{j+1}}$.
(h) $w_{s_i} + hd < w_{s_j} < w_{s_{j+1}} \le w_{s_{i+1}} + hd$.
(i) $w_{s_i} + hd < w_{s_j} \le w_{s_{i+1}} + hd < w_{s_{j+1}}$.

From Fig. A3, we can obtain the formula below to determine α_{ij}^h for the cases with $h < 0$:

$$\alpha_{ij}^h = \begin{cases} -\infty & \text{for case (a)} \\ r_i + \left[w_{s_j} - \left(w_{s_{i+1}} + hd\right)\right]/\lambda & \text{for cases (b), (e)} \\ r_i - \mu_j - \left[w_{s_{i+1}} + hd - w_{s_j}\right]/\nu & \text{for cases (c), (g)} \\ \mu_i + \left[w_{s_j} - \left(w_{s_i} + hd\right)\right]/\nu & \text{for cases (d), (h)} \\ \mu_i - \left[w_{s_i} + hd - w_{s_j}\right]/\nu - \mu_j & \text{for case (f)} \\ \mu_i + \left[w_{s_j} - \left(w_{s_i} + hd\right)\right]/\nu & \text{for case (i)} \\ \quad + \max\{0, \eta_i - \mu_j\} & \end{cases} \tag{A3}$$

It can be seen that the cases in Figs. A2 and A3 have a one-to-one correspondence.

For $h > 0$ and $h < 0$, exchanging the indexes i and j and changing the sign of h in Figs. A2 and A3, we can get the cases for determining β_{ij}^h values and the following relationship between the formulae of α_{ij}^h and β_{ij}^h:

$$\beta_{ij}^h = -\alpha_{ji}^{-h}.$$

In summary, for all situations where $h = 0$, $h > 0$, and $h < 0$, α_{ij}^h values can be calculated using Equations (A1), (A2), and (A3), respectively. The cases and values of β_{ij}^h can be obtained using the relationships between α and β.

Biographies

Yun Jiang is a software engineer at Alcatel-Lucent, Cambridge, UK. He received his Ph.D. in Industrial Engineering and Engineering Management from Hong Kong University of Science and Technology (HKUST) and his B.S. in Automatic Control from Hua Zhong University of Science and Technology in China. Before joining Alcatel-Lucent, he held research and academic positions at HKUST, National University of Singapore, Bilkent University in Turkey, and Warwick University in the UK. His primary research interests are in scheduling and planning in production and logistics. He has published research papers in *IEEE Transactions*, *IIE Transactions*, and *Operations Research*.

Jiyin Liu is Professor of Operations Management in the School of Business and Economics at Loughborough University, UK. He previously taught at Northeastern University and The Hong Kong University of Science and Technology in China. He received his Ph.D. in Manufacturing Engineering and Operations Management from The University of Nottingham in the UK and his B.Eng. in Industrial Automation and M.Eng. in Systems Engineering from Northeastern University in China. His research interests are in operations planning and scheduling in production and logistics, as well as in modeling, analysis, and solution of practical operations problems. His research has been published in various academic journals such as *European Journal of Operational Research*, *IEEE Transactions*, *IIE Transactions*, *International Journal of Production Research*, *Journal of the Operational Research Society*, *Naval Research Logistics*, *Operations Research*, and *Transportation Research*.