Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2008, Article ID 364279, 21 pages
doi:10.1155/2008/364279

Research Article

Optimal Scheduling of Material Handling Devices in a PCB Production Line: Problem Formulation and a Polynomial Algorithm

Ada Che[1] and Chengbin Chu[2]

[1] *School of Management, Northwestern Polytechnical University, Xi'an 710072, China*
[2] *ISTIT, Université de Technologie de Troyes, BP 2060, 12 Rue Marie Curie, 10010 Troyes Cedex, France*

Correspondence should be addressed to Ada Che, ache@nwpu.edu.cn

Received 30 January 2006; Revised 22 October 2007; Accepted 15 March 2008

Recommended by Jerzy Warminski

Modern automated production lines usually use one or multiple computer-controlled robots or hoists for material handling between workstations. A typical application of such lines is an automated electroplating line for processing printed circuit boards (PCBs). In these systems, cyclic production policy is widely used due to large lot size and simplicity of implementation. This paper addresses cyclic scheduling of a multihoist electroplating line with constant processing times. The objective is to minimize the cycle time, or equivalently to maximize the production throughput, for a given number of hoists. We propose a mathematical model and a polynomial algorithm for this scheduling problem. Computational results on randomly generated instances are reported.

1. Introduction

Modern automated production lines usually use one or multiple computer-controlled robots or hoists for material handling between workstations. A typical example is an automated electroplating line for processing printed circuit boards (PCBs). Such a production line usually consists of a loading station, a sequence of chemical tanks, an unloading station, and a crew of identical programmable hoists, as shown in Figure 1. Parts to be processed enter the system from the loading station, and then are processed successively through tanks, and finally leave the system from the unloading station. Each tank contains chemicals required for a specific electroplating step in the processing of parts such as acid cleaning, acid activating, copper plating, rinsing, and so on. Each tank can process only one part at a time. The processing time in a tank may be a given constant or allowed to vary within a given window. Due to

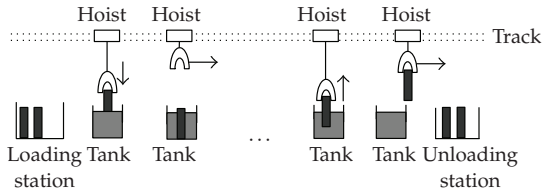

Figure 1: An automated multihoist electroplating line for processing PCBs.

specific characteristics of chemical treatment, as soon as the processing operation of a part is completed in a tank, it must be immediately removed from that tank and transported to the next one without any delay. Otherwise, defective parts may be produced due to oxidization and contamination. In an automated electroplating line, the movements of parts between the tanks are performed by a crew of computer-controlled hoists on a single track. By optimizing the sequence and start times of the hoist moves, we can optimize the throughput of the line. This problem is commonly known as the hoist scheduling problem in the literature [1–9].

Due to large lot size in electroplating operations, the production is often organized in a cyclic manner, and only one part type is processed repeatedly in the line in a production period. In such a cyclic production system, the hoists are programed to perform a fixed sequence of moves repeatedly. Each repetition of the sequence is called a *cycle*. The duration of a cycle is called the *cycle time* or the *cycle length*. Normally, one raw part enters and one finished part leaves the line within a cycle. The throughput rate is the inverse of the cycle time. Therefore, minimizing the cycle time is equivalent to maximizing the throughput of a production line.

This paper addresses cyclic scheduling of a multihoist electroplating line with constant processing times, that is, the processing times of a part in tanks are constants. This type of scheduling problem arises typically from high-precision electroplating systems, in which the quality of treatment of a part mainly depends on its processing times in tanks. In literature, some studies (e.g., [1, 3–5, 7, 8, 10–12]) deal with the scheduling problem with time windows, that is, the processing time of parts in each tank must fall into a given time window. This problem is NP-hard both for the single-hoist case and for the multihoist case. Hence, the researchers proposed heuristics or branch-and-bound algorithms for the single-hoist or multihoist scheduling problem with time windows. It should be noted that the polynomial algorithm developed in this paper for the problem with constant processing times can be served as a heuristic for the problem with time windows.

For the scheduling problem with constant processing times, Agnetis [13] developed polynomial algorithms for lines with two or three tanks and a single hoist for material handling. The same problem for any given number of tanks was shown to be solvable in polynomial time by Levner et al. [14]. Che and Chu [6] extended Levner's work and developed an efficient algorithm for single-hoist electroplating lines with multifunctional and/or duplicate tanks. As the number of tanks increases, material handling between the tanks often becomes bottlenecks. To eliminate such bottlenecks and increase the throughput, it is a common practice to use more than one hoist in an electroplating line with more than 10 tanks. Karzanov and Livshits [15] appear to be the first authors to study the cyclic multihoist scheduling problem. They studied the system with *parallel* tracks (i.e., hoists travel along their respective tracks) and proposed an $O(N^3)$ algorithm to find the minimal number of hoists for a given cycle time, where N is the number of tanks in a production line. Kats and Levner [16]

extended their results and found that the problem of minimizing the number of hoists for all possible cycle times can be solved in $O(N^5)$ time. Kats and Levner [17] also developed an $O(N^3 \log N)$ algorithm for the multihoist scheduling problem for a given hoist assignment.

In the above studies, the researchers assumed that the hoists travel along their respective parallel tracks and, therefore, the collision avoidance among the hoists was not addressed. However, almost all practical electroplating lines have only one available track. This paper addresses the single-track, multihoist scheduling problem with constant processing times. When the hoists travel along a common track, the problem is much more complicated than that with parallel tracks. With parallel tracks, it is not required to address collision avoidance constraints among the hoists and the problem can be reduced to either an assignment problem or a simple variant of a single-hoist problem if the hoist assignment is given. However, for electroplating lines with a single track, we must address the collision avoidance constraints among the hoists either on the track or on the tanks. As will be shown in this paper, the solution of the problem with a single track differs from that for parallel tracks. Liu and Jiang [18] proposed an efficient algorithm for the single-track, two-hoist scheduling problem with constant processing times. In this paper, we develop a mathematical model and a corresponding polynomial algorithm for the single-track, multihoist scheduling problem with constant processing times.

2. Problem formulation

Consider an electroplating line consisting of a loading station M_0, N chemical tanks, $M_1, M_2, \ldots, M_N$, and an unloading station M_{N+1}. The stations or tanks are arranged in a row from left to right in the following order: $M_0, M_1, \ldots, M_N, M_{N+1}$, as shown in Figure 2. A single type of parts is to be processed in the line. The part flow can be described as follows. After a part is removed from M_0, it is processed successively through tanks $M_1, M_2, \ldots, M_N$ and finally leaves the system from M_{N+1}. Each tank can process at most one part at a time and the processing time in M_i is a given constant t_i. There are K identical hoists on a single track, which are responsible for transporting parts between the tanks. Without loss of generality, we assume that the hoists are numbered, from left to right, from 0 to $K-1$, as shown in Figure 2. For simplicity, the hoist movement of transporting a part from M_i to M_{i+1} is called *move* i, $0 \le i \le N$. Each move i consists of three simple hoist operations: (1) lift up a part from M_i; (2) transport the part to M_{i+1}; and (3) lower the part onto M_{i+1}. The time required for the hoists to perform move i, $i = 0, 1, \ldots, N$, is θ_i, where lifting up a part from M_i, $i = 0, 1, \ldots, N$, and lowering a part onto M_i, $i = 1, 2, \ldots, N+1$, take v_i and μ_i, respectively. The hoist movement without transporting any part is called a *void move*. The time for the hoists to perform a void move from M_i to M_j, $i, j = 0, 1, \ldots, N+1$, is $d_{i,j}$. $d_{i,j}$'s satisfy the triangular inequality. Finally, let constant δ be the allowable minimum distance among the hoists on the track in order to avoid collision, that is, if the distance between two hoists is less than δ, then a collision happens between them. For simplicity of notation, δ is measured in time in the remainder, which is equal to the allowable minimum distance divided by the travel speed of the hoists.

The hoists are programed to perform a fixed sequence of moves repeatedly. The sequence of moves performed by the hoists during a cycle is called a *cyclic schedule*. Our objective is to find a cyclic hoist schedule such that the cycle time T is minimized. A cyclic hoist schedule consists of the set of moves performed by each hoist and their respective starting times relative to the start of the cycle. To define a hoist schedule, let Y_i be the starting time of move i relative

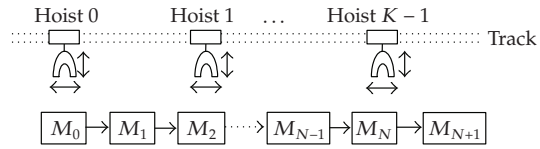

Figure 2: Part flow through an electroplating line with N chemical tanks and K hoists.

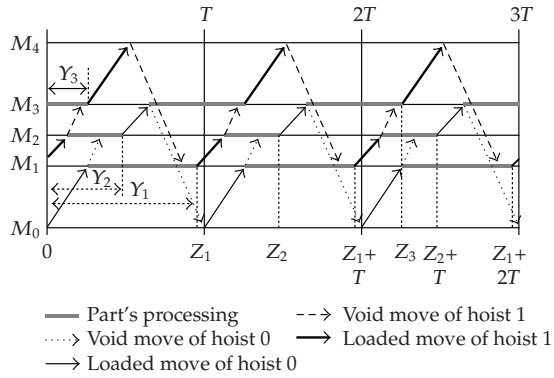

Figure 3: A cyclic schedule with three tanks and two hoists.

to the start of a cycle, $i = 0, 1, \ldots, N$, and r_i be the index of the hoist to perform move i (i.e., move i is performed by hoist r_i), $r_i \in \{0, 1, \ldots, K-1\}$, $i = 0, 1, \ldots, N$. Thus, a cyclic schedule can be uniquely defined by $(T, \{r_i,\ i = 0, 1, \ldots, N\}, \{Y_i,\ i = 0, 1, \ldots, N\})$.

Figure 3 illustrates a cyclic schedule for an electroplating line with three chemical tanks (not including the loading station M_0 and the unloading station M_4) and two hoists for material handling. Three complete cycles are illustrated in Figure 3. Without loss of generality, we assume that $Y_0 = 0$, that is, move 0 happens at the start of a cycle which also implies that a part is introduced into the system at the start of a cycle. Note that if $Y_0 > 0$, we can change the origin of the time axis such that $Y_0 = 0$. From Figure 3, we see that parts are introduced into the system at time instant $0, T, 2T, \ldots$. Each hoist performs a fixed sequence of moves cyclically. For this example, we have $r_0 = 0$, $r_1 = 1$, $r_2 = 0$, $r_3 = 1$, that is, hoist 0 performs moves 0 and 2 cyclically, while hoist 1 executes moves 1 and 3 repeatedly. From Figure 3, a cyclic schedule is uniquely defined by $(T, \{r_i, i = 0, 1, \ldots, N\}, \{Y_i,\ i = 0, 1, \ldots, N\})$.

Our objective is to find a cyclic hoist schedule denoted by $(T, \{r_i, i = 0, 1, \ldots, N\}, \{Y_i,\ i = 0, 1, \ldots, N\})$ such that the cycle time T is minimized. A cyclic schedule $(T, \{r_i, i = 0, 1, \ldots, N\}, \{Y_i,\ i = 0, 1, \ldots, N\})$ is feasible if and only if it satisfies the following four families of constraints:

(i) processing time constraints. The parts' processing time in M_i is exactly t_i, $i = 1, 2, \ldots, N$;

(ii) tank capacity constraints. Each tank can process at most one part at a time;

(iii) hoist availability constraints. There is no conflict in the use of the same hoist between any pair of moves executed by that hoist, since any hoist cannot perform two moves at the same time;

(iv) collision-free constraints. The hoists travel on a single track and no collisions happen during a cycle.

As mentioned above, at the start of each cycle, a part is introduced into the system from the loading station. This means that the parts are introduced into the line at time instant $0, T, 2T, \ldots$, as shown in Figure 3. For the sake of simplicity, the part introduced into the system at time nT $(n \geq 0)$ is called part n. With this definition, move i of part n represents the hoist movement of transporting part n from M_i to M_{i+1}. Let Z_j be the completion time of the jth processing operation of part 0, which is also the starting time of move j of part 0. From Figure 3, Z_j can be calculated by using the following formula:

$$Z_j = \sum_{i=1}^{j} (\theta_{i-1} + t_i), \quad j = 1, 2, \ldots, N, \tag{2.1}$$

with $Z_0 = 0$. By definition, $Z_j + nT$ is the completion time of the jth processing operation of part n and also the starting time of move j of part n, for any $0 \leq j \leq N$, for any $n \geq 0$. In steady state, the starting time of move j within $[0, T)$ (i.e., relative to the start of a cycle) is given by

$$Y_j = Z_j \bmod T, \quad j = 0, 1, \ldots, N. \tag{2.2}$$

Figure 3 illustrates the above relationship between Y_j and Z_j.

In the following, we formulate our problem using the notion of prohibited intervals of the cycle time, which was first introduced by Levner et al. [14] into the cyclic scheduling of no-wait systems. Note that the part processing time requirements are implicitly taken into account by using (2.1) and (2.2) to compute the starting times of the moves, as soon as T is known.

2.1. Formulation of the tank capacity constraints

The tank capacity constraints require that the processing of any two successive parts on the same tank cannot be overlapped, since each tank can process one part at a time. Furthermore, by taking into account the times required for lifting up a part from a tank and lowering a part onto a tank, we must have

$$T \geq t_j + \mu_j + \nu_j, \quad \forall\, 1 \leq j \leq N. \tag{2.3}$$

This relation leads to

$$T \geq \beta = \max_{1 \leq j \leq N} (t_j + \mu_j + \nu_j). \tag{2.4}$$

2.2. Formulation of the hoist availability constraints

The hoist availability constraints require that there is no conflict in the use of the same hoist between any pair of moves executed by that hoist. This implies that there must be sufficient time interval between the start of any two moves performed by the same hoist, since any hoist cannot perform two moves at the same time. This means that, for any pair of moves (j, i), $j = 0, \ldots, N-1$, $i = j+1, \ldots, N$, if move i and move j are performed by the same hoist

(i.e., $r_i = r_j$), then move i of *any part* must be executed either sufficiently before or sufficiently after move j of *any part*. Due to the cyclic nature of the problem, it is sufficient to consider the hoist availability constraints for move i of part 0 and move j of part n (for any $n \geq 1$) if move i and move j are performed by the same hoist. Therefore, for any pair of moves (j,i), $j = 0,1,\ldots,N-1$, $i = j+1,\ldots,N$, such that $r_i = r_j$, move i of part 0 must be done either sufficiently before or sufficiently after move j of part n, for any $n = 1,2,\ldots$.

Appendix A shows that when part n enters the system, for any $n \geq n^* + 1$, where $n^* = \min(N+K, \lceil (Z_N + \theta_N)/\beta \rceil) - 1$, part 0 must have left the system. Hence, there is no more conflict in the use of the hoist between part 0 and part n when $n \geq n^* + 1$. So, we need to consider only those n's such that $n = 1,2,\ldots,n^*$. In fact, n^* is the upper bound on the number of parts simultaneously processed in a production line.

Figure 4(a) shows the case when move i of part 0 happens after move j of part n, while Figure 4(b) shows the case when move i of part 0 is done before move j of part n. By definition, move i of part 0 starts at Z_j and ends at $Z_i + \theta_i$, and move j of part n starts at $Z_j + nT$ and ends at $Z_j + nT + \theta_j$, as shown in Figures 4(a) and 4(b). It follows from Figure 4(a) that

$$Z_i \geq Z_j + nT + \theta_j + d_{j+1,i}, \tag{2.5}$$

where $d_{j+1,i}$ is the time for the hoist to travel, upon completion of move j of part n, from M_{j+1} to M_i to perform move i of part 0. Similarly, it follows from Figure 4(b) that

$$Z_j + nT \geq Z_i + \theta_i + d_{i+1,j}, \tag{2.6}$$

where $d_{i+1,j}$ is the time for the hoist to travel, upon completion of move i of part 0, from M_{i+1} to M_j to perform move j of part n.

To simplify the notation, define $f_{ij} \equiv Z_i - Z_j + \theta_i + d_{i+1,j}$. According to (2.5) and (2.6), in any case, we must have

$$\begin{aligned} &\text{either } nT \leq Z_i - Z_j - \theta_j - d_{j+1,i} = -f_{j,i}, \\ &\text{or } nT \leq Z_i - Z_j + \theta_i + d_{i+1,j} = f_{i,j}, \\ &\forall\, 1 \leq n \leq n^*, \quad \forall\, 0 \leq j \leq N-1, \quad j+1 \leq i \leq N \quad \text{such that } r_i = r_j. \end{aligned} \tag{2.7}$$

Equations (2.7) are equivalent to

$$nT \notin \left(-f_{j,i}, f_{i,j}\right), \quad \forall\, 1 \leq n \leq n^*,\ \forall\, 0 \leq j \leq N-1,\ j+1 \leq i \leq N \text{ such that } r_i = r_j. \tag{2.8}$$

Example 2.1. An electroplating line consists of three tanks, that is, $N = 3$. There are two hoists available in the system, that is, $K = 2$. The processing times are as follows: $t_1 = 16$, $t_2 = 8$, $t_3 = 14$. For all $0 \leq i < j \leq 4$, the time for a void move from M_i to M_j is obtained by using $d_{i,j} = d_{j,i} = \sum_{k=i}^{j-1} d_{k,k+1}$ with $d_{0,1} = d_{3,4} = 4$, $d_{1,2} = d_{2,3} = 2$. For all $0 \leq i \leq 4$, $\mu_i = 0.5$, $v_i = 0.5$. The times required for executing moves: $\theta_0 = \theta_3 = 6$, $\theta_1 = \theta_2 = 4$. We set $\delta = 1$. For this example, according to (2.1), we have $Z_0 = 0$, $Z_1 = 22$, $Z_2 = 34$, $Z_3 = 52$. By definition, $\beta = \max_{1 \leq j \leq N}(t_j + \mu_j + v_j) = 17$, $n^* = \min(3+2, \lceil (Z_3 + \theta_3)/\beta \rceil) - 1 = 3$, $f_{1,0} = 32$, $f_{2,0} = 46$, $f_{3,0} = 70$, $f_{2,1} = 20$, $f_{3,1} = 44$, $f_{3,2} = 30$, $f_{0,1} = -16$, $f_{0,2} = -26$, $f_{0,3} = -42$, $f_{1,2} = -8$, $f_{1,3} = -24$, $f_{2,3} = -14$.

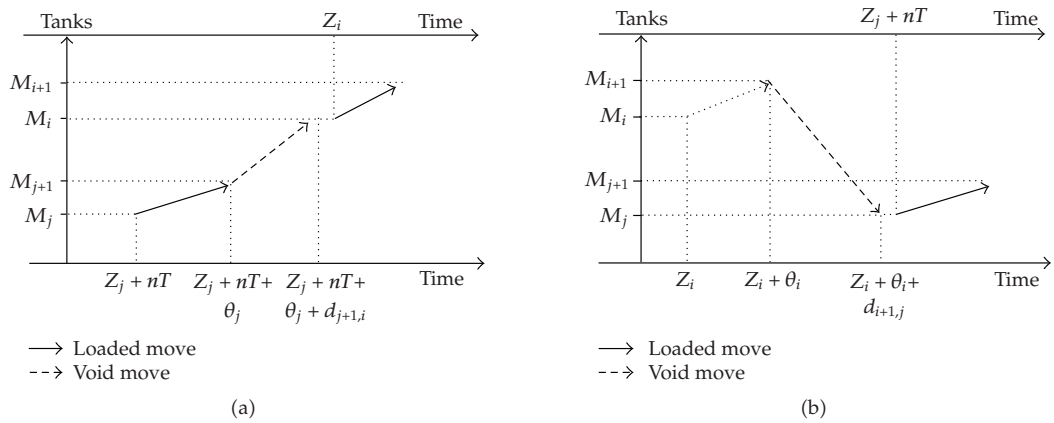

Figure 4: (a) Hoist availability constraint when move i succeeds move j ($r_i = r_j$). (b) Hoist availability constraint when move i precedes move j ($r_i = r_j$).

2.3. Formulation of the collision-free constraints

In this subsection, we formulate the collision-free constraints among the hoists. This is accomplished by considering possible collisions in the execution of moves. For any two moves i and j executed by *different* hoists, if their execution requires that the hoists use a common zone of the track, then either move i must sufficiently precede move j or move j must sufficiently precede move i. Otherwise, possible collisions between the hoists may happen, since they use a common zone of the track at the same time. In the remainder of the paper, for any two moves i and j, without loss of generality, we assume that $i > j$. Three cases should be considered.

Case 1. $r_i > r_j$ and $i > j + 1$, see Figure 5(a). In this case, in view of the part flow shown in Figure 2, no collisions will happen between the two hoists during their execution of moves i and j.

Case 2. $r_i > r_j$ and $i = j + 1$, see Figure 5(b). In this case, hoist r_j and hoist r_i may collide at M_i, onto which a part is lowered by hoist r_j and from which another part is lifted up by hoist r_i.

Case 3. $r_i < r_j$, see Figure 5(c). In this case, hoists r_i and r_j will use an overlapping zone of the track from M_j to M_{i+1} in order to execute moves i and j, and collisions may happen between them when passing through this overlapping zone.

From this analysis, when there are multiple hoists on a single track, collisions may happen among hoists not only when they use an overlapping zone of the track, but also when using the same tank, from which a part is lifted up by one hoist and onto which another part is lower down by another hoist. Such a tank is called a *boundary tank* in this paper. In the following, we will first address Case 3 and then consider Case 2.

2.3.1. Execution of two moves requires using an overlapping zone of the track

By Case 3 (see Figure 5(c), hoists r_i and r_j will use an overlapping zone of the track from M_j to M_{i+1} in order to execute moves i and j, and collisions may happen between them. In order to

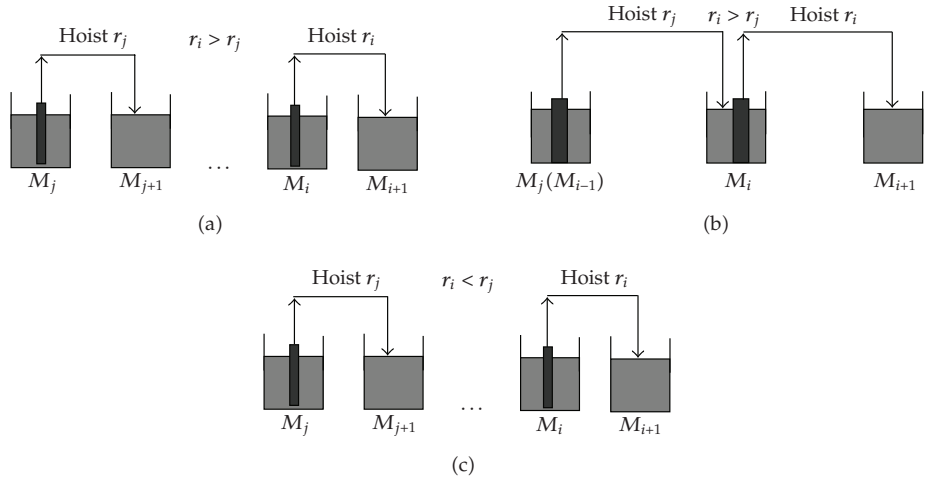

Figure 5: (a) No collisions in the execution of two moves ($r_i > r_j,\ i > j+1$). (b) Execution of two moves requires using a boundary tank M_i ($r_i > r_j,\ i = j+1$). (c) Execution of two moves requires using an overlapping zone of the track from M_j to M_{i+1} ($r_i < r_j,\ i > j$).

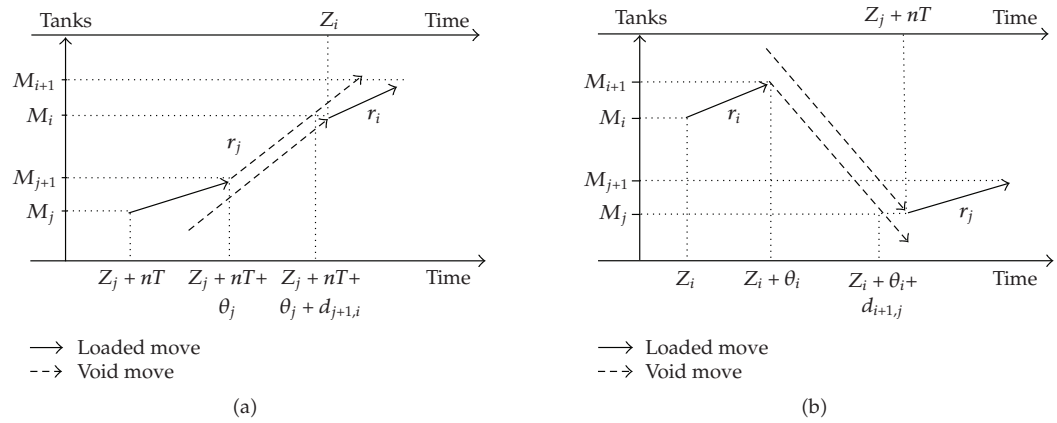

Figure 6: (a) Collision-free constraint when move i succeeds move j ($r_i < r_j$). (b) Collision-free constraint when move i precedes move j ($r_i < r_j$).

avoid collision between hoists r_i and r_j such that $r_i < r_j$, they cannot use this overlapping zone at the same time. There must be sufficient time interval between them in using this overlapping zone. This means that, for any pair of moves $(j,i),\ j = 0,1,\ldots,N-1,\ i = j+1,\ldots,N$, such that $r_i < r_j$, move i of part 0 must be done either sufficiently before or sufficiently after move j of part n, for any $n = 1,2,\ldots,n^*$.

Figure 6(a) shows the case when move i of part 0 is done after move j of part n, that is, $Z_i > Z_j + nT$. In order to avoid collision, after hoist r_j finishes move j of part n, it should arrive at M_i before hoist r_i and should move to a higher position than M_i (i.e., move to a position nearer to the unloading station than M_i), as shown in Figure 6(a). Note that the earliest time at which hoist r_j arrives at M_i is $Z_j + nT + \theta_j + d_{j+1,i}$, and the latest time at which hoist r_i arrives

at M_i is Z_i, and the allowable minimum distance among the hoists is δ. Hence, as shown in Figure 6(a), the following constraint must be satisfied:

$$Z_i \geq Z_j + nT + \theta_j + d_{j+1,i} + (r_j - r_i)\delta. \tag{2.9}$$

Similarly, as shown in Figure 6(b), if move i of part 0 is done before move j of part n, that is, $Z_i < Z_j + nT$, we must have

$$Z_j + nT \geq Z_i + \theta_i + d_{i+1,j} + (r_j - r_i)\delta. \tag{2.10}$$

From (2.9) and (2.10), we have

$$\begin{aligned} &\text{either } nT \leq Z_i - Z_j - \theta_j - d_{j+1,i} - (r_j - r_i)\delta = -f_{j,i} - (r_j - r_i)\delta, \\ &\text{or } nT \geq Z_i - Z_j + \theta_i + d_{i+1,j} + (r_j - r_i)\delta = f_{i,j} + (r_j - r_i)\delta, \\ &\forall\, 1 \leq n \leq n^*, \quad \forall\, 0 \leq j \leq N-1, \quad j+1 \leq i \leq N \quad \text{such that } r_i < r_j. \end{aligned} \tag{2.11}$$

The constraints (2.11) can be equivalently written as

$$nT \notin \left(-f_{j,i} - (r_j - r_i)\delta, f_{i,j} + (r_j - r_i)\delta\right), \quad \forall\, 1 \leq n \leq n^*,\ \forall\, 0 \leq j \leq N-1,\ j+1 \leq i \leq N \text{ such that } r_i < r_j. \tag{2.12}$$

2.3.2. Execution of two moves requires using a boundary tank

By Case 2 (see Figure 5(b)), hoists r_j and r_i may collide at M_i, onto which a part is lowered by hoist r_j and from which another part is lifted up by hoist r_i. In order to avoid collision, part m (for any $m \geq 0$) must have been lifted up from M_i, for any $1 \leq i \leq N$, when part $m+1$ arrives at M_i. Note that part m will leave M_i at time $Z_i + mT + \nu_i$, and part $m+1$ will arrive at M_i at time $Z_{i-1} + (m+1)T + \theta_{i-1} - \mu_i$. knowing that the allowable minimum distance among the hoists is δ, in order to avoid collision between hoists r_j and r_i, we must have

$$Z_{i-1} + (m+1)T + \theta_{i-1} - \mu_i \geq Z_i + mT + \nu_i + (r_i - r_{i-1})\delta, \quad \forall\, 1 \leq i \leq N \text{ such that } r_{i-1} < r_i. \tag{2.13}$$

This relation leads to

$$T \geq t_i + \mu_i + \nu_i + (r_i - r_{i-1})\delta, \quad \forall\, 1 \leq i \leq N \text{ such that } r_{i-1} < r_i. \tag{2.14}$$

This relation can be equivalently written as

$$T \notin \left(-\infty, t_i + \mu_i + \nu_i + (r_i - r_{i-1})\delta\right), \quad \forall\, 1 \leq i \leq N \text{ such that } r_{i-1} < r_i, \tag{2.15}$$

From the above formulation of the problem, we see that the collision-free constraints can be formulated as (2.12) and (2.15). It should be emphasized that this formulation of the collision-free constraints is complete, since by Cases 1, 2, and 3 (or Figures 5(a), 5(b), and 5(c)) possible combinations of r_i and r_j are all taken into account for any pair of moves (j, i), $j = 0, 1, \ldots, N-1$, $i = j+1, \ldots, N$. Note that (2.8) and (2.12) can be generalized as

$$nT \notin \left(-f_{j,i} - (r_j - r_i)\delta f_{i,j} + (r_j - r_i)\delta\right), \quad \forall\, 1 \leq n \leq n^*,\ \forall\, 0 \leq j \leq N-1,\ j+1 \leq i \leq N \text{ such that } r_i \leq r_j. \tag{2.16}$$

According to (2.4), (2.15), and (2.16), the multihoist electroplating line scheduling problem considered in this paper can be formulated as the following prohibited intervals for the cycle time T:

$$\text{Minimize } T, \tag{2.17}$$

subject to (2.4), (2.15), and (2.16).

Note that (2.4), (2.15), and (2.16) can be equivalently written as

$$\begin{aligned} T \notin \mathbf{Q}(R) &\equiv (-\infty,\beta) \cup \left\{ \bigcup_{\substack{1\le i\le N \\ r_{i-1}<r_i}} \left(-\infty, t_i+\mu_i+\nu_i+(r_i-r_{i-1})\delta\right) \right\} \\ &\cup \left\{ \bigcup_{0\le j\le N} \bigcup_{\substack{j+1\le i\le N \\ r_i\le r_j}} \left\{ \left(-f_{j,i}-(r_j-r_i)\delta, f_{i,j}+(r_j-r_i)\delta\right) \cup \left(\frac{-f_{j,i}-(r_j-r_i)\delta}{2}, \frac{f_{i,j}+(r_j-r_i)\delta}{2}\right) \right.\right. \\ &\qquad \left.\left. \cup \cdots \cup \left(\frac{-f_{j,i}-(r_j-r_i)\delta}{n^*}, \frac{f_{i,j}+(r_j-r_i)\delta}{n^*}\right) \right\} \right\}, \end{aligned} \tag{2.18}$$

where vector $R = (r_0, r_1, \ldots, r_N)$. R is called the hoist assignment in the remainder. It can be found from (2.18) that $\mathbf{Q}(R)$ is a union of R-parameterized open prohibited intervals for the cycle time T. For a given R, $\mathbf{Q}(R)$ is a union of open prohibited intervals for the cycle time T.

In this Section, we formulate our problem as a series of prohibited intervals for the cycle time, that is, $\mathbf{Q}(R)$. Each family of the problem constraints (i.e., tank capacity constraints, hoist availability constraints, and collision-free constraints) corresponds to a set of prohibited intervals for T, for example, the hoist availability constraint between a pair of moves (j,i) such that $r_i = r_j$ corresponds to a set of prohibited intervals $nT \notin (-f_{j,i}, f_{i,j})$, for $n = 1, 2, \ldots, n^*$. Due to this property, for a given hoist assignment R, if $T \notin \mathbf{Q}(R)$, then such a T must be *feasible*, since, by definition, such a T falls into no prohibited intervals in $\mathbf{Q}(R)$, and consequently all problem constraints must be satisfied.

3. Problem analysis

In this Section, we perform a property analysis for the mathematical model we developed in the above Section. Based on this analysis, we will show that the optimal cycle time for the problem is necessarily one of special values of the cycle time. Hence, the optimal cycle time can be found by detecting the feasibility for each one of these special values of the cycle time, and our problem is thus reduced to a feasibility checking problem for a given value of T.

Theorem 3.1. *Given a hoist assignment R, the optimal cycle time $T^*(R) \in \mathbf{A}(R)$, where*

$$\begin{aligned} \mathbf{A}(R) \equiv {} & \{\beta\} \cup \left\{ x \mid x = t_i+\mu_i+\nu_i+(r_i-r_{i-1})\delta,\ x>\beta,\ 1\le i\le N,\ r_{i-1}<r_i \right\} \\ & \cup \left\{ y \mid y = \frac{f_{i,j}+(r_j-r_i)\delta}{n},\ y>\beta,\ 1\le n\le n^*,\ 0\le j\le N-1,\ j+1\le i\le N,\ r_i\le r_j \right\}. \end{aligned} \tag{3.1}$$

Proof. Given a hoist assignment R, $\mathbf{Q}(R)$ is a union of open prohibited intervals which are not necessarily disjoint. However, $\mathbf{Q}(R)$ can be considered as a union of *disjoint* prohibited intervals after merging of the intersecting ones. Therefore, $\mathbf{Q}(R)$ can be rewritten as

$$\mathbf{Q}(R) = (a_1, b_1) \cup (a_2, b_2) \cup \cdots \cup (a_H, b_H) \tag{3.2}$$

with $-\infty = a_1 < b_1 \le a_2 < \cdots < b_{i-1} \le a_i < b_i \le a_{i+1} < \cdots < b_H$. □

Example 1 (continued)

If $R = (r_0, r_1, r_2, r_3) = (0, 1, 0, 1)$, that is, move 0 and move 2 are performed by hoist 0 while move 1 and move 3 are executed by hoist 1, according to (2.18), the following relation holds

$$\begin{aligned}
T \notin \mathbf{Q}(R) &= (-\infty, \beta) \cup \left\{ \left(-\infty, t_1 + \mu_1 + \nu_1 + \delta \right) \cup \left(-\infty, t_3 + \mu_3 + \nu_3 + \delta \right) \right\} \\
&\cup \left\{ \left(-f_{0,2}, f_{2,0} \right) \cup \left(\frac{-f_{0,2}}{2}, \frac{f_{2,0}}{2} \right) \cup \left(\frac{-f_{0,2}}{3}, \frac{f_{2,0}}{3} \right) \right\} \\
&\cup \left\{ \left(-f_{1,2} - \delta, f_{2,1} + \delta \right) \cup \left(\frac{-f_{1,2} - \delta}{2}, \frac{f_{2,1} + \delta}{2} \right) \cup \left(\frac{-f_{1,2} - \delta}{3}, \frac{f_{2,1} + \delta}{3} \right) \right\} \\
&\cup \left\{ \left(-f_{1,3}, f_{3,1} \right) \cup \left(\frac{-f_{1,3}}{2}, \frac{f_{3,1}}{2} \right) \cup \left(\frac{-f_{1,3}}{3}, \frac{f_{3,1}}{3} \right) \right\} \\
&= (-\infty, 17) \cup \{(-\infty, 18) \cup (-\infty, 16)\} \cup \{(26, 46) \cup (13, 23) \cup (8.66, 15.33)\} \\
&\cup \{(7, 21) \cup (3.5, 10.5) \cup (2.33, 7)\} \cup \{(24, 44) \cup (12, 22) \cup (8, 14.66)\}.
\end{aligned} \tag{3.3}$$

After merging of the intersecting prohibited intervals in $\mathbf{Q}(R)$, we have $T \notin \mathbf{Q}(R) = (-\infty, 23) \cup (24, 46)$. Hence, $a_1 = -\infty$, $b_1 = 23$, $a_2 = 24$, $b_2 = 46$.

We can easily find that the optimal cycle time $T^*(R)$ is necessarily the upper bound of the first open prohibited interval, that is, b_1, since b_1 is the smallest cycle time that is not prohibited by $\mathbf{Q}(R)$. Note that the upper bound of any *disjoint* prohibited interval, after merging of the intersecting intervals, is necessarily an upper bound of one of the prohibited intervals before merging of the intersecting intervals. This means that b_1 is necessarily one of the upper bounds of the prohibited intervals before merging of the intersecting ones. Therefore, it follows from (2.18) that $b_1 \in \mathbf{A}(R)$. Thus, we have Theorem 3.1.

Corollary 3.2. *For any hoist assignment R, $\mathbf{A}(R) \subset \mathbf{A}_T$ always holds where*

$$\begin{aligned}
\mathbf{A}_T \equiv {} & \{\beta\} \cup \left\{ \hat{x} \mid \hat{x} = t_i + \mu_i + \nu_i + k\delta,\ \hat{x} > \beta,\ 1 \le i \le N,\ 1 \le k \le K - 1 \right\} \\
& \cup \left\{ \hat{y} \mid \hat{y} = \frac{f_{i,j} + k\delta}{n},\ \hat{y} > \beta,\ 0 \le j \le N - 1,\ j + 1 \le i \le N,\ 0 \le k \le K - 1,\ 1 \le n \le n^* \right\}.
\end{aligned} \tag{3.4}$$

The correctness of Corollary 3.2 is straightforward, since we have $(r_i - r_{i-1}) \in \{1, 2, \ldots, K-1\}$ for any $1 \le i \le N$ such that $r_{i-1} < r_i$, and $(r_j - r_i) \in \{0, 1, \ldots, K-1\}$ for any $0 \le j \le N-1$, $j+1 \le i \le N$ such that $r_i \le r_j$.

By Theorem 3.1 and Corollary 3.2, we have the following corollary.

Corollary 3.3. *The optimal cycle time T^* for the problem $T^* \in \mathbf{A}_T$.*

By Corollary 3.3, the optimal cycle time T^* can be found by detecting the feasibility for each value of the cycle time in $\mathbf{A}_T$ in increasing order until the first feasible cycle time is found, which is the optimal cycle time for the problem. Our problem is thus reduced to the feasibility checking problem for a given value of T.

Example 1 (continued)

The set $\mathbf{A}_T = \{17, 18, 21, 22, 22.5, 23, 23.5, 30, 31, 32, 33, 35, 35.5, 44, 45, 46, 47, 70, 71\}$. The optimal cycle time for the problem can be found by checking the feasibility of these values of the cycle time in increasing order.

4. Feasibility checking for a given value of T

As mentioned in Section 2, for a given hoist assignment R, if $T \notin \mathbf{Q}(R)$, then such a T must be *feasible*. Hence, given a value of T, say T_0, T_0 must be *feasible* if there exists a hoist assignment R such that $T_0 \notin \mathbf{Q}(R)$. Such an R is accordingly called a feasible hoist assignment for T_0. Thus, to check feasibility for T_0, we only need to check whether there exists a feasible hoist assignment R for T_0. If so, then T_0 is feasible. Our basic idea is to first derive sufficient and necessary constraints that R must satisfy in order that $T_0 \notin \mathbf{Q}(R)$. By solving the derived constraints for R, we then detect the feasibility for T_0 and obtain a feasible hoist assignment R if T_0 is feasible.

Theorem 4.1. *For a given cycle time T_0, in order that $T_0 \notin \mathbf{Q}(R)$, a sufficient and necessary condition is that R satisfies the following constraints:*

$$r_i - r_{i-1} \le k - 1, \quad \text{if } T_0 \in \left(-\infty, t_i + \mu_i + \nu_i + k\delta\right), \ \forall 1 \le k \le K-1, \ \forall 1 \le i \le N, \tag{4.1}$$

$$r_j - r_i \le k-1, \quad \text{if } \left(s_{i,j}^k - 1\right)T_0 \in \left(-f_{j,i} - k\delta, f_{i,j} + k\delta\right), \ \forall 0 \le j \le N-1, \ j+1 \le i \le N, \ 0 \le k \le K-1, \tag{4.2}$$

$$r_i - r_0 \ge 0, \quad \forall 1 \le i \le N, \tag{4.3}$$

$$r_i - r_0 \le K - 1, \quad \forall 1 \le i \le N, \tag{4.4}$$

where $s_{i,j}^k = \lceil (f_{i,j} + k\delta)/T_0 \rceil$, where $\lceil x \rceil$ is the smallest integer greater than or equal to x.

Proof. Note that $T_0 \notin \mathbf{Q}(R)$ is equivalent to (2.4), (2.15), and (2.16) hold for T_0. Constraint (2.4) means that if $T_0 < \beta$, then T_0 must be infeasible and no further feasibility checking is needed. Hence, in the remainder, without loss of generality, we assume that $T_0 \ge \beta$. With this assumption, (2.4) is always satisfied. In the following, we will first derive the sufficient and necessary constraints that R must satisfy in order that (2.15) holds for T_0 *(Part A)*, and then we will derive the sufficient and necessary constraints that R must satisfy in order that (2.16) holds for T_0 *(Part B)*. □

Part A. The sufficiency and necessity of (4.1) in order that (2.15) holds for T_0.

We first derive the necessary constraints that R must satisfy in order that (2.15) holds for T_0. Since $1 \le r_i - r_{i-1} \le K - 1$ for any $1 \le i \le N$ such that $r_{i-1} < r_i$, in order that (2.15) holds for T_0, the following relation must hold

$$r_i - r_{i-1} < k, \quad \text{if } T_0 \in \left(-\infty, t_i + \mu_i + \nu_i + k\delta\right), \ \forall 1 \le k \le K-1, \ \forall 1 \le j \le N. \tag{4.5}$$

The correctness of (4.5) is clear, since if (4.5) was not satisfied, then we have $T_0 \in (-\infty, t_i + \mu_i + \nu_i + k\delta)$ and $r_i - r_{i-1} \geq k$, for some $1 \leq k \leq K-1$ and $1 \leq i \leq N$, and consequently (2.15) will be violated for T_0. Since r_i takes only integer, (4.5) is equivalent to (4.1). Hence, in order that (2.15) holds for T_0, R must satisfy (4.1).

We now prove the sufficiency of (4.1) in order that (2.15) holds for T_0. Assume that (2.15) does NOT hold for T_0 when (4.1) holds. We will show that this assumption will lead to contradictory facts and thus is incorrect. By assumption, since (2.15) does NOT hold for T_0, there must exists an i such that $r_{i-1} < r_i$ and $T_0 \in (-\infty, t_i + \mu_i + \nu_i + (r_i - r_{i-1})\delta)$. By letting $m = r_i - r_{i-1}$, we have $T_0 \in (-\infty, t_i + \mu_i + \nu_i + m\delta)$ for this specified i. Since, by assumption, (4.1) holds and $T_0 \in (-\infty, t_i + \mu_i + \nu_i + m\delta)$, according to (4.1), we must have $r_i - r_{i-1} \leq m - 1$. This is in contradiction with the fact that $m = r_i - r_{i-1}$. This means that the assumption that (2.15) does NOT hold for T_0 when (4.1) holds is incorrect. We, thus, prove the sufficiency of (4.1) in order that (2.15) holds for T_0.

Part B. The sufficiency and necessity of (4.2) in order that (2.16) holds for T_0.

We first derive the necessary constraints that R must satisfy in order that (2.16) holds for T_0. Since $0 \leq r_j - r_i \leq K-1$, for any $0 \leq j \leq N-1$, $j+1 \leq i \leq N$ such that $r_i \leq r_j$, in order that (2.16) holds for T_0, the following relation must hold

$$\begin{aligned} r_j - r_i < k, \quad &\text{if there exists an } n,\ 1 \leq n \leq n^* \text{ such that } nT_0 \in \left(-f_{j,i} - k\delta, f_{i,j} + k\delta\right) \\ &\forall\, 0 \leq j \leq N-1,\ j+1 \leq i \leq N,\ 0 \leq k \leq K-1. \end{aligned} \tag{4.6}$$

The correctness of (4.6) is clear, since if (4.6) was not satisfied, then we have $nT_0 \in (-f_{j,i} - k\delta, f_{i,j} + k\delta)$ and $r_j - r_i \geq k$, for some $1 \leq n \leq n^*$, $0 \leq k \leq K-1$, $0 \leq j \leq N-1$, $j+1 \leq i \leq N$, and consequently (2.16) will be violated for T_0. As shown in Appendix B, checking whether there exists an n, $1 \leq n \leq n^*$ such that $nT_0 \in (-f_{j,i} - k\delta, f_{i,j} + k\delta)$, for all $0 \leq j \leq N-1$, $j+1 \leq i \leq N$, $0 \leq k \leq K-1$ is equivalent to checking whether $(s_{i,j}^k - 1)\, T_0 \in (-f_{j,i} - k\delta, f_{i,j} + k\delta)$, where $s_{i,j}^k = \lceil (f_{i,j} + k\delta)/T_0 \rceil$. As a result, (4.6) can be equivalently expressed as

$$r_j - r_i < k, \quad \text{if } \left(s_{i,j}^k - 1\right) T_0 \in \left(-f_{j,i} - k\delta, f_{i,j} + k\delta\right),\ \forall\, 0 \leq j \leq N-1,\ j+1 \leq i \leq N,\ 0 \leq k \leq K-1. \tag{4.7}$$

Since r_j and r_i take only integers, (4.7) is equivalent to (4.2). Hence, in order that (2.16) holds for T_0, R must satisfy (4.2).

We now prove the sufficiency of (4.2) in order that (2.16) holds for T_0. Similarly, assume that (2.16) does NOT hold for T_0 when (4.2) holds. We will also show that this assumption will lead to contradictory facts and thus is incorrect. By assumption, since (2.16) does NOT hold for T_0, we must have $nT_0 \in (-f_{j,i} - (r_j - r_i)\delta, f_{i,j} + (r_j - r_i)\delta)$ for some $1 \leq n \leq n^*$, $0 \leq j \leq N-1$, $j+1 \leq i \leq N$, such that $r_i \leq r_j$. By letting $m = r_j - r_i$, we have $nT_0 \in (-f_{j,i} - m\delta, f_{i,j} + m\delta)$ for the specified n, i, and j. Since, by assumption, (4.2) holds and $nT_0 \in (-f_{j,i} - m\delta, f_{i,j} + m\delta)$ for the specified n, i, and j, according to (4.2), we must have $r_j - r_i \leq m - 1$. This is in contradiction with the fact that $m = r_j - r_i$. This means that the assumption that (2.16) does NOT hold for T_0 when (4.2) holds is incorrect. We, thus, prove the sufficiency of (4.2) in order that (2.16) holds for T_0.

Since there are K hoists in the production line, we must have

$$0 \leq r_i \leq K-1, \quad \forall\, 0 \leq i \leq N. \tag{4.8}$$

With the line configuration in Figure 2, it is easy to find that move 0 must be performed by hoist 0 in order to avoid collision among the hoists. Thus, without loss of generality, we assume that $r_0 = 0$. With this assumption, constraints (4.8) can be equivalently written as (4.3) and (4.4).

To sum up, in order that $T_0 \notin \mathbf{Q}(R)$, a sufficient and necessary condition is that R satisfies the constraints (4.1)–(4.4).

Example 1 (continued)

We illustrate the feasibility checking for $T_0 = 23$. According to (4.1)–(4.4), in order that $T_0 \notin \mathbf{Q}(R)$, a sufficient and necessary condition is that R satisfies the following constraints:

$$r_0 - r_1 \leq -1, \tag{4.9}$$

$$r_0 - r_2 \leq 0, \tag{4.10}$$

$$r_0 - r_3 \leq -1, \tag{4.11}$$

$$r_2 - r_3 \leq -1, \tag{4.12}$$

$$r_i - r_0 \leq 1, \quad \forall\, 1 \leq i \leq 3. \tag{4.13}$$

We illustrate how to derive (4.12). By definition, we derive $s^0_{3,2} = 2$, $s^1_{3,2} = 2$. Hence, $(s^0_{3,2} - 1)T_0 = 23 \in (-f_{2,3}, f_{3,2}) = (14, 30)$; $(s^1_{3,2} - 1)T_0 = 23 \in (-f_{2,3} - \delta, f_{3,2} + \delta) = (13, 31)$. This leads to $r_2 - r_3 \leq -1$ and $r_2 - r_3 \leq 0$ according to (4.2). The latter relation is redundant. We, thus, derive (4.12). The other constraints can be derived in a similar way. Note that the constraints $r_i - r_0 \geq 0$, for all $1 \leq i \leq 3$, are redundant with the consideration of (4.9)–(4.11) and thus were not shown here.

By Theorem 4.1, (4.1)–(4.4) are the sufficient and necessary constraints that R must satisfy in order that $T_0 \notin \mathbf{Q}(R)$. Note that each constraint in (4.1)–(4.4) can be equivalently written in the form of $r_j - r_i \geq c_{ij}$, where c_{ij} is an integer, $0 \leq i, j \leq N$. Due to this special structure, solving (4.1)–(4.4) can be transformed into a longest-path problem in a directed graph, as will be described in detail. In a directed graph $G(V, E)$, where V and E are, respectively, the set of vertices and the set of arcs, a weight $w(e)$ is associated with each arc $e \in E$. Let $h(e)$ and $t(e)$ be, respectively, the head and the tail of arc $e \in E$ (i.e., arc e goes from vertex $t(e)$ to vertex $h(e)$). Let π_v denote the potential of vertex $v \in V$. Thus, each arc e represents a constraint $\pi_{h(e)} - \pi_{t(e)} \geq w(e)$.

The directed graph constructed from (4.1)–(4.4) contains $N + 1$ vertices, $0, 1, \ldots, N$, the potentials of which are, respectively, $r_0, r_1, \ldots, r_N$. Each constraint in the form of $r_j - r_i \geq c_{i,j}$, $0 \leq i, j \leq N$ from (4.1)–(4.4) is represented by an arc from vertex i to vertex j with weight $c_{i,j}$. Thus, the set of constraints in (4.1)–(4.4) can be represented as

$$r_j - r_i \geq c_{i,j}, \quad \forall\, (i, j) \in E. \tag{4.14}$$

With the constructed directed graph, a hoist assignment R satisfies (4.1)–(4.4) if and only if all the arcs in graph $G(V, E)$ satisfy (4.14). Let C be a directed cycle (circuit) on graph $G(V, E)$, then by (4.14)

$$\sum_{\forall\,(i,j)\in C} (r_j - r_i) \geq \sum_{\forall\,(i,j)\in C} c_{i,j}. \tag{4.15}$$

Since $\sum_{\forall (i,j)\in C}(r_j - r_i) = 0$, we have

$$\sum_{\forall (i,j)\in C} c_{i,j} \le 0. \tag{4.16}$$

Thus, if there are positive circuits in the associated directed graph, then (4.1)–(4.4) are said to be infeasible in sense that they cannot lead to any feasible hoist assignment. On the other hand, if there is no positive circuit in the directed graph, then (4.1)–(4.4) are said to be feasible or self-consistent and a feasible hoist assignment can be derived from them. The following theorem gives how to derive a feasible hoist assignment R if (4.1)–(4.4) are feasible or self-consistent and its complexity.

Theorem 4.2. *For a given cycle time T_0, if there is no positive circuit in the directed graph constructed from* (4.1)–(4.4)*, then a feasible hoist assignment R must exist and it can be obtained in $O(N^3)$ in the worst case.*

Proof. Let l_i be the length of the longest path from vertex 0 to vertex i on graph $G(V, E)$ satisfying

$$l_k + \left\| P_{k,i} \right\| \le l_i, \quad 0 \le i,\ k \le N,\ k \ne i, \tag{4.17}$$

where $P_{k,i}$ denotes the longest path from vertex k to vertex i. By definition, $l_0 = 0$. It is easy to see that $R = (r_0, r_1, \ldots, r_N) = (l_0, l_1, \ldots, l_N)$ satisfies (4.14), which also implies that such an R satisfies (4.1)–(4.4). By Theorem 4.1, such an R is a feasible hoist assignment for T_0. Thus, solving (4.1)–(4.4) and consequently checking the feasibility for T_0 can be transformed into solving a longest path problem for the corresponding directed graph, which can be solved in $O(|V||E|)$, where $|V|$ and $|E|$ are, respectively, the number of vertices and the number of arcs in the directed graph. For this problem, we have $|V| = N + 1$, $|E| \le (N + 1)(N + 2)/2$. Hence, we have Theorem 4.2. □

Example 1 (continued)

The directed graph corresponding to (4.9)–(4.13) is shown in Figure 7. By solving the longest path problem for this graph, we obtain $l_0 = 0$, $l_1 = 1$, $l_2 = 0$, $l_3 = 1$. We, thus, find a feasible hoist assignment $R = (r_0, r_1, r_2, r_3) = (0, 1, 0, 1)$ for $T_0 = 23$. This implies that $T_0 = 23$ is feasible. It can be shown that the values of the cycle time less than $T_0 = 23$ in $\mathbf{A}_T$ are all infeasible. Therefore, $T_0 = 23$ is the optimal cycle time for the problem. From (2.2), we have $Y_0 = 0$, $Y_1 = 22$, $Y_2 = 11$, $Y_3 = 6$. The optimal cyclic schedule for $T_0 = 23$ is shown in Figure 3. We see from Figure 3 that hoists 0 and 1 use the overlapping zone from M_1 to M_3. We also see that there is sufficient time interval between them in entering into the overlapping zone. Hence, possible collisions between them are avoided.

5. Algorithm and complexity analysis

The process to solve the hoist scheduling problem considered in this paper can be summarized as follows.

Step 1. Calculate the completion time Z_j, for all $j = 0, 1, \ldots, N$, according to (2.1).

Step 2. Calculate $f_{i,j}$ for all $0 \le j \le N-1$, $j+1 \le i \le N$, and β.

Step 3. Construct set $\mathbf{A}_T$.

Step 4. Sort all the values of the cycle time in $\mathbf{A}_T$ in increasing order. Check the feasibility for each given value of T, say T_0, in $\mathbf{A}_T$ in increasing order as follows.

Step 4.1. Calculate $s_{i,j}^k$, for all $0 \le j \le N-1$, $j+1 \le i \le N$, $0 \le k \le K-1$.

Step 4.2. Using (4.1)–(4.4) to derive the sufficient and necessary constraints that R must satisfy in order that $T_0 \notin \mathbf{Q}(R)$.

Step 4.3. Construct the directed graph and solve the corresponding longest path problem. If T_0 is feasible, obtain the corresponding R, and go to Step 5. Otherwise, go to Step 4.1 to check the feasibility for the next value of the cycle time in $\mathbf{A}_T$.

Step 5. Compute the move starting times for the optimal cycle time by using (2.2).

Theorem 5.1. *The multihoist scheduling problem with constant processing times is solvable in $O(N^6K)$ time in the worst case.*

Proof. Steps 1 and 2, respectively, require $O(N)$ and $O(N^2)$ times. By the definition of $\mathbf{A}_T$, the total number of the values of the cycle time in $\mathbf{A}_T$ is $O(N^2Kn^*)$. Hence, Step 3 can be implemented in $O(N^2Kn^*)$ time. In Step 4, sorting all the values of the cycle time in $\mathbf{A}_T$ in increasing order requires $O(N^2Kn^*(\log N + \log K + \log n^*))$ time.

For each given T_0, Step 4.1 can be done in $O(N^2K)$ time. Step 4.2 can be implemented in $O(N^2K)$. By Theorem 4.2, Step 4.3 can be implemented in $O(N^3)$ in the worst case. Without loss of generality, we assume that each hoist in the line must perform at least one move and each move can be performed by one and only one hoist. With this assumption, there are at most $(N+1)$ hoists really used for material handling in a production line with N processing tanks. Hence, we assume that $K \le (N+1)$. Thus, for each given T_0, Steps 4.1–4.3 can be done in $O(N^3)$. It is known that the total number of the values of the cycle time in $\mathbf{A}_T$ is $O(N^2Kn^*)$. To sum up, the algorithm runs in $O(N^5Kn^*)$ time.

Note that $n^* = \min(N+K, \lceil (Z_N + \theta_N)/\beta \rceil) - 1$. This leads to $n^* \le (N+K) \le 2N+1$. Therefore, the algorithm runs in $O(N^6K)$ time in the worst case. □

6. Computational results

The proposed algorithm was encoded in C++. In this section, we first use an example to verify the correctness of the proposed algorithm. Randomly generated instances are then used to further evaluate the performance of the algorithm. The computational experiment was done on a PC with a Pentium IV 3.0 GHz processor.

The example has 20 processing tanks numbered from 1 to 20, stations 0 and 21 being the loading station and the unloading station, respectively. The processing times for tanks 1 to 20 are 160, 180, 90, 150, 200, 190, 290, 170, 290, 230, 240, 86, 180, 300, 240, 180, 310, 200, 170, and 70, respectively. For any i such that $0 \le i \le N$, the time for a void move from M_i to M_{i+1}, that is, $d_{i,i+1} = 3$. The other void move times are obtained accordingly with $d_{i,j} = d_{j,i} = \sum_{k=i}^{j-1} d_{k,k+1}$, where $0 \le i < j \le N+1$. The move times $\theta_i = d_{i,i+1} + 20$, where $v_i = \mu_i = 10$ for any $0 \le i \le N$.

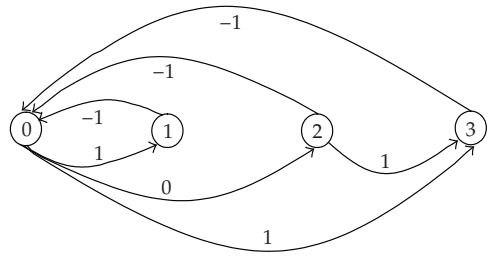

Figure 7: The directed graph corresponding to (4.9)–(4.13).

Table 1: Optimal multihoist cycle times for the example.

Number of hoists	Proposed algorithm		B and B algorithm	
	T^*	CPU (s)	T^*	CPU (s)
$K = 1$	2316	0.015	2316	0.2
$K = 2$	1160	0.078	1160	24.6
$K = 3$	628	0.106	628	143.6
$K = 4$	358	0.109	358	59.8
$K = 5$	344	0.156	344	153.1

Note that the problem with constant processing times, as addressed in this study, can be considered as a special case of the problem with time windows, that is, zero-width time window. Hence, we can also solve the example using the branch-and-bound algorithm proposed in [5]. Table 1 gives the computational results for the example using the proposed algorithm and the branch-and-bound algorithm in [5]. In Table 1, the columns with T^* provide with the optimal multihoist cycle times for the example, while the columns with CPU represent their corresponding computation times (measured in seconds).

It can be observed from Table 1 that the optimal multihoist cycle times obtained with the proposed polynomial algorithm are the same as those obtained with the branch-and-bound algorithm in [5]. However, our polynomial algorithm significantly outperforms the branch-and-algorithm regarding computation times. This is due to the fact that the algorithm developed in this study exploits specific properties to the problem with constant processing times. The optimal 4-hoist schedule for the example is shown in Figure 8.

In addition, randomly generated instances were used to evaluate the performance of the proposed algorithm. The test instances were generated as follows using integer uniform distributions. For any i such that $0 \leq i \leq N$, the time for a void move from M_i to M_{i+1}, that is, $d_{i,i+1}$, was randomly generated on [2, 5]. The other void move times are obtained accordingly with $d_{i,j} = d_{j,i} = \sum_{k=i}^{j-1} d_{k,k+1}$, where $0 \leq i < j \leq N + 1$. The move times θ_i is set to $\theta_i = d_{i,i+1} + 20$, where $\nu_i = \mu_i = 10$ for any i such that $0 \leq i \leq N$. The processing times t_i were generated on [30, 300]. We fix the value of $N = 20, 30, 40$, and 50. For each $N = 20, 30, 40$, and 50, 100 test instances were generated. We consider five values of K : 1, 2, 3, 4, and 5, and 2000 ($= 4 \times 100 \times 5$) test problems were solved.

Table 2 reports the average reduction in optimal cycle times (or equivalently improvement on the throughput) achieved by the multihoist schedules with respect to the single-hoist schedules, calculated by the optimal single-hoist cycle time divided by the optimal multihoist

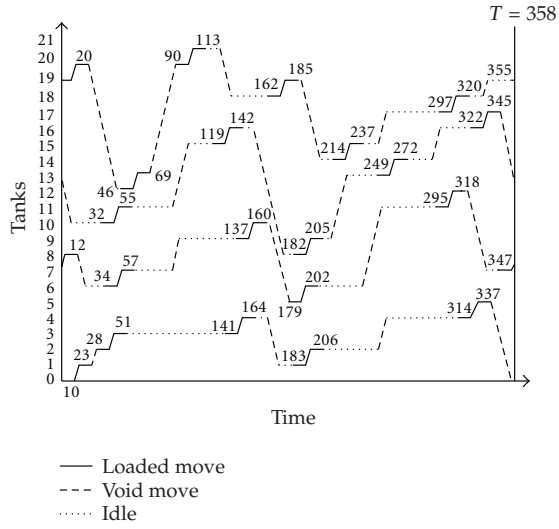

Figure 8: Optimal 4-hoist schedule for the example.

Table 2: Average reduction in cycle times by using multihoist for test instances.

Problem	$K = 2$	$K = 3$	$K = 4$	$K = 5$
$N = 20$	2.08	3.98	6.25	7.63
$N = 30$	2.26	4.39	6.81	9.37
$N = 40$	2.10	3.89	6.28	8.79
$N = 50$	2.18	3.81	6.06	8.67

cycle time. We see from Table 2 that the reduction in optimal cycle times achieved by the multihoist schedules is very significant, ranging from 2.08 to 9.37.

Table 3 gives the average computation times in CPU seconds for test instances. It can be seen from Table 3 that the algorithm developed in this study is very efficient. Note that practical electroplating lines generally have 10 to 20 processing tanks and 1 to 3 hoists. For this size of problems, the computation times are within 80 milliseconds. For the 20-tank and 30-tank instances with 1 to 5 hoists, our algorithm found optimal cycle times generally within one second. Even for the very large problem with 50 tanks and 5 hoists, the proposed algorithm can obtain the optimal solution within 25 seconds.

A further examination of Table 3 reveals that the computation times increase with the number of tanks and hoists, as the algorithm runs in $O(N^6K)$ time in the worst case according to its complexity estimation, but the computation times increase not so rapidly with the number of tanks as the complexity of the algorithm indicates. This may be due to the fact that the complexity analysis is a worst-case estimation and sometimes may be far from the reality. To find the optimal cycle time, the values of the cycle time in $\mathbf{A}_T$ are checked in increasing order until a feasible one is found. In the complexity estimation, the number of values of the cycle time to be checked is estimated at $O(N^3K)$ in the worst case. This estimation is too pessimistic. In fact, only a small part of the values of the cycle time in $\mathbf{A}_T$ needs to be checked before the optimal cycle time is found, and more importantly, the total number of values of the cycle time in $\mathbf{A}_T$ is significantly less than its theoretical estimation $O(N^3K)$ (less than 10% for most of the

Table 3: Average computation times for test instances (s).

Problem	$K = 1$	$K = 2$	$K = 3$	$K = 4$	$K = 5$
$N = 20$	0.01	0.05	0.08	0.10	0.14
$N = 30$	0.05	0.22	0.57	0.90	1.30
$N = 40$	0.13	0.62	1.98	4.11	6.26
$N = 50$	0.28	1.39	4.46	11.90	23.92

instances), since most values of $\hat{x}$ and $\hat{y}$ in $\mathbf{A}_T$ are less than β and the values of $\hat{x}$ and/or $\hat{y}$ may take the same value.

7. Conclusion

This paper proposed a polynomial algorithm for the no-wait cyclic multihoist scheduling problem in an electroplating line. Computational results on randomly generated instances have shown that the algorithm is very efficient. The algorithm developed in this study can be served as a subroutine or a heuristic in solving the problem with time windows.

An important extension of this study is to develop efficient heuristic for the hoist scheduling problem with time windows, which is an NP-hard problem, using the algorithm developed in this study as a subroutine. To achieve this extension, we should take the processing times, which must be within given time windows, as decision variables of our problem. If all processing times are fixed, then the corresponding problem can be solved using the proposed algorithm. Thus, the key to solving the problem with time windows is to design efficient search strategy to find an optimal combination of fixed processing times within given time windows. This is our ongoing work.

Appendices

A. Upper bound on the number of parts simultaneously processed in a production line

It is known that part 0 arrives at the unloading station M_{N+1} at time $Z_N + \theta_N$ and part n is introduced into the system at time nT. In order that when part n enters the system, part 0 has already left the system, it is *sufficient* that

$$nT \geq Z_N + \theta_N. \tag{A.1}$$

It follows from (2.4) that

$$\lceil (Z_N + \theta_N)/\beta \rceil T \geq \lceil Z_N + \theta_N \rceil \geq Z_N + \theta_N. \tag{A.2}$$

This means that (A.1) must hold for $n \geq \lceil (Z_N + \theta_N)/\beta \rceil$.

On the other hand, since the K hoists have to perform all the moves within a cycle, we must also have

$$T \geq \sum_{i=0}^{N} \theta_i / K. \tag{A.3}$$

It follows from (2.4) and (A.3) that

$$(N+K)T \geq N\beta + \sum_{i=0}^{N}\theta_i \geq \sum_{i=1}^{N} t_i + \sum_{i=0}^{N}\theta_i = Z_N + \theta_N. \tag{A.4}$$

Hence, (A.1) must hold for $n \geq (N+K)$.

To sum up, when part n enters the system, for any $n \geq n^* + 1$, where $n^* = \min(N + K, \lceil (Z_N + \theta_N)/\beta \rceil) - 1$, part 0 must have left the system; n^* can be understood as the upper bound on the number of parts simultaneously processed in a production line.

B. Equivalence of (4.6) and (4.7)

In constraints (4.6), for all $0 \leq j \leq N-1$, $j+1 \leq i \leq N$, $0 \leq k \leq K-1$, we must check whether there exists an n, $1 \leq n \leq n^*$, such that $nT_0 \in (-f_{j,i} - k\delta, f_{i,j} + k\delta)$. This requires that $nT_0 \in (-f_{j,i} - k\delta, f_{i,j} + k\delta)$ be checked from $n = 1$ to n^*. Hence, $nT_0 \in (-f_{j,i} - k\delta, f_{i,j} + k\delta)$ will be checked for at most n^* times. In fact, this is not necessary.

By definition, $s_{i,j}^k = \lceil (f_{i,j} + k\delta)/T_0 \rceil$, where $\lceil x \rceil$ is the smallest integer greater than or equal to x. This implies that $s_{i,j}^k$ is smallest integer such that $s_{i,j}^k T_0 \geq f_{i,j} + k\delta$. Hence, for any $s_{i,j}^k \leq n \leq n^*$, we always have $nT_0 \geq f_{i,j} + k\delta$, that is, $nT_0 \notin (-f_{j,i} - k\delta, f_{i,j} + k\delta)$ for any $s_{i,j}^k \leq n \leq n^*$. On the other hand, since $s_{i,j}^k$ is the smallest integer n such that $nT_0 \geq f_{i,j} + k\delta$, we must have $(s_{i,j}^k - 1)T_0 < f_{i,j} + k\delta$. Hence, we check whether $(s_{i,j}^k - 1)T_0 \leq -f_{j,i} - k\delta$.

Case 1. if $(s_{i,j}^k - 1)T_0 \leq -f_{j,i} - k\delta$, then for any $1 \leq n < s_{i,j}^k - 1$, we also have $nT_0 \leq -f_{j,i} - k\delta$. As a result, in this case, we have $nT_0 \notin (-f_{j,i} - k\delta, f_{i,j} + k\delta)$ for any $1 \leq n \leq n^*$.

Case 2. if $(s_{i,j}^k - 1)T_0 > -f_{j,i} - k\delta$, we find an $n = s_{i,j}^k - 1$ such that $nT_0 \in (-f_{j,i} - k\delta, f_{i,j} + k\delta)$.

From this analysis, for all $0 \leq j \leq N-1$, $j+1 \leq i \leq N$, $0 \leq k \leq K-1$, checking whether there exists an n, $1 \leq n \leq n^*$, such that $nT_0 \in (-f_{j,i} - k\delta, f_{i,j} + k\delta)$ is equivalent to checking whether $(s_{i,j}^k - 1)T_0 \in (-f_{j,i} - k\delta, f_{i,j} + k\delta)$.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China, under Grant no. 50605052 and the Program for New Century Excellent Talents in Universities of Ministry of Education, China, under Grant no. NCET-06-0875.

References

[1] L. W. Phillips and P. S. Unger, "Mathematical programming solution of a hoist scheduling program," *AIIE Transactions*, vol. 8, no. 2, pp. 219–225, 1976.

[2] W. Song, Z. B. Zabinsky, and R. L. Storch, "An algorithm for scheduling a chemical processing tank line," *Production Planning & Control*, vol. 4, no. 4, pp. 323–332, 1993.

[3] L. Lei and T. L. Wang, "Determining optimal cyclic hoist schedules in a single-hoist electroplating line," *IIE Transactions*, vol. 26, no. 2, pp. 25–33, 1994.

[4] H. Chen, C. Chu, and J.-M. Proth, "Cyclic scheduling of a hoist with time window constraints," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 1, pp. 144–152, 1998.

[5] A. Che and C. Chu, "Single-track multi-hoist scheduling problem: a collision-free resolution based on a branch and bound approach," *International Journal of Production Research*, vol. 42, no. 12, pp. 2435–2456, 2004.
[6] A. Che and C. Chu, "A polynomial algorithm for no-wait cyclic hoist scheduling in an extended electroplating line," *Operations Research Letters*, vol. 33, no. 3, pp. 274–284, 2005.
[7] J. Leung and G. Zhang, "Optimal cyclic scheduling for printed-circuit-board production lines with multiple hoists and general processing sequences," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 3, pp. 480–484, 2003.
[8] J. M. Y. Leung, G. Zhang, X. Yang, R. Mak, and K. Lam, "Optimal cyclic multi-hoist scheduling: a mixed integer programming approach," *Operations Research*, vol. 52, no. 6, pp. 965–976, 2004.
[9] M. Dawande, H. N. Geismar, S. P. Sethi, and C. Sriskandarajah, "Sequencing and scheduling in robotic cells: recent developments," *Journal of Scheduling*, vol. 8, no. 5, pp. 387–426, 2005.
[10] R. Armstrong, S. Gu, and L. Lei, "A greedy algorithm to determine the number of transporters in a cyclic electroplating process," *IIE Transactions*, vol. 28, no. 5, pp. 347–355, 1996.
[11] L. Lei, R. Armstrong, and S. Gu, "Minimizing the fleet size with dependent time window and single-track constraints," *Operations Research Letters*, vol. 14, no. 2, pp. 91–98, 1993.
[12] C. Varnier, A. Bachelu, and P. Baptiste, "Resolution of the cyclic multi-hoists scheduling problem with overlapping partitions," *Information Systems and Operations Research*, vol. 35, no. 4, pp. 277–284, 1997.
[13] A. Agnetis, "Scheduling no-wait robotic cells with two and three machines," *European Journal of Operational Research*, vol. 123, no. 2, pp. 303–314, 2000.
[14] E. Levner, V. Kats, and V. E. Levit, "An improved algorithm for cyclic scheduling in a robotic cell," *European Journal of Operational Research*, vol. 97, no. 3, pp. 500–508, 1997.
[15] A. V. Karzanov and E. M. Livshits, "Minimal quantity of operators for serving a homogeneous linear technological process," *Automation and Remote Control*, vol. 39, pp. 445–450, 1978.
[16] V. Kats and E. Levner, "Minimizing the number of robots to meet a given cyclic schedule," *Annals of Operations Research*, vol. 69, pp. 209–226, 1997.
[17] V. Kats and E. Levner, "Cyclic scheduling in a robotic production line," *Journal of Scheduling*, vol. 5, no. 1, pp. 23–41, 2002.
[18] J. Liu and Y. Jiang, "An efficient optimal solution to the two-hoist no-wait cyclic scheduling problem," *Operations Research*, vol. 53, no. 2, pp. 313–327, 2005.

Special Issue on
Singular Boundary Value Problems for Ordinary Differential Equations

Call for Papers

The purpose of this special issue is to study singular boundary value problems arising in differential equations and dynamical systems. Survey articles dealing with interactions between different fields, applications, and approaches of boundary value problems and singular problems are welcome.

This Special Issue will focus on any type of singularities that appear in the study of boundary value problems. It includes:

- Theory and methods
- Mathematical Models
- Engineering applications
- Biological applications
- Medical Applications
- Finance applications
- Numerical and simulation applications

Before submission authors should carefully read over the journal's Author Guidelines, which are located at http://www.hindawi.com/journals/bvp/guidelines.html. Authors should follow the Boundary Value Problems manuscript format described at the journal site http://www.hindawi.com/journals/bvp/. Articles published in this Special Issue shall be subject to a reduced Article Processing Charge of €200 per article. Prospective authors should submit an electronic copy of their complete manuscript through the journal Manuscript Tracking System at http://mts.hindawi.com/ according to the following timetable:

Manuscript Due	May 1, 2009
First Round of Reviews	August 1, 2009
Publication Date	November 1, 2009

Lead Guest Editor

Juan J. Nieto, Departamento de Análisis Matemático, Facultad de Matemáticas, Universidad de Santiago de Compostela, Santiago de Compostela 15782, Spain;
juanjose.nieto.roig@usc.es

Guest Editor

Donal O'Regan, Department of Mathematics, National University of Ireland, Galway, Ireland;
donal.oregan@nuigalway.ie