Operations Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

An Efficient Optimal Solution to the Two-Hoist No-Wait Cyclic Scheduling Problem

Jiyin Liu, Yun Jiang,

To cite this article:
Jiyin Liu, Yun Jiang, (2005) An Efficient Optimal Solution to the Two-Hoist No-Wait Cyclic Scheduling Problem. Operations Research 53(2):313-327. https://doi.org/10.1287/opre.1040.0167

Operations Research
Vol. 53, No. 2, March–April 2005, pp. 313–327
issn 0030-364X | eissn 1526-5463 | 05 | 5302 | 0313

informs®

doi 10.1287/opre.1040.0167
© 2005 INFORMS

An Efficient Optimal Solution to the Two-Hoist No-Wait Cyclic Scheduling Problem

Jiyin Liu
Business School, Loughborough University, Loughborough, Leicestershire LE11 3TU, United Kingdom, j.y.liu@lboro.ac.uk

Yun Jiang
Department of Industrial Engineering, Bilkent University, 06800 Bilkent, Ankara, Turkey, jiangyun@bilkent.edu.tr

Hoist scheduling is a typical problem in the operation of electroplating systems. The cyclic scheduling policy is widely used in these systems in industry. Research on hoist scheduling has focused on the cyclic problem to minimize the cycle length. Most previous studies consider the single-hoist case. In practice, however, more than one hoist is often used in an electroplating line. This paper addresses the two-hoist, no-wait cyclic scheduling problem, in which the tank-processing times are constants and, upon completion of processing in a tank, the parts have to be moved to the next tank immediately. Based on the analysis of the problem properties, a polynomial algorithm is developed to obtain an optimal schedule. This algorithm first identifies a set of thresholds, which are special values of the cycle length, so that the feasibility property may change only at these thresholds. Feasibility checking is then carried out on each individual threshold in ascending order. The first feasible threshold found will be the optimal cycle length, and the corresponding feasible schedule is an optimal hoist schedule.

Subject classifications: production/scheduling: cyclic; two hoists; no-wait; noncrossing; manufacturing: automated electroplating systems; robotic cells.
Area of review: Optimization.
History: Received February 2002; revisions received October 2002, July 2003, November 2003; accepted December 2003.

1. Introduction

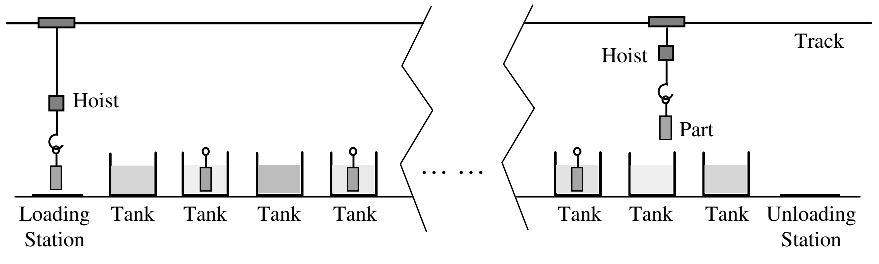
This paper is motivated by the practical problem of scheduling material-handling hoists in electroplating systems. Electroplating is a necessary process in producing printed circuit boards (PCB), which are widely used in computers, telecommunication equipment, and many other electronics products. An electroplating system is a production line with a series of processing tanks that contain the required chemical solutions. Parts to be processed must visit a given sequence of tanks according to the technological requirements. Normally, only one part type is processed repeatedly in the line in a production period. The required processing time in any one tank is therefore identical for all these parts. The processing time in a tank may be fixed or restricted to vary within a given window. One or more hoists mounted on a common track are used to transfer the parts between the tanks. An example of such an electroplating line is illustrated in Figure 1.

In practice, electroplating lines operate cyclically. Usually one part enters into the line, and another leaves the line after completing all the required processing steps, in each cycle. The term "one part" is used here for simplicity of description. In many systems, the actual "part" may be a unit load of parts in a carrier. The duration of a cycle is called the cycle length. Each hoist repeats a sequence of part movements in every cycle. Hoist scheduling allocates the hoists to perform all the required moves in a cycle to maximize the production throughput, i.e., to minimize the cycle length. In some studies, an r-part cycle is considered where r parts are introduced into the line and another r parts are completed and taken out of the line during a longer cycle. Although this can potentially further increase throughput in theory, it is seldom used in practice as it complicates production supervision and hoist control.

1.1. Problem Statement

In this paper, we study the two-hoist, no-wait cyclic scheduling problem. We consider an electroplating system consisting of a loading station, n processing stations (chemical tanks), and an unloading station. The stations are arranged in a line from left to right in the following order: the loading station (station 0), the processing stations (stations $1, 2, \ldots, n$), and the unloading station (station $n+1$). The position of station i is w_i, $i = 0, 1, \ldots, n+1$. In some systems, loading and unloading are performed at the same station. In this case, station $n+1$ does not exist physically and $w_{n+1} \equiv w_0$. Each station can process at most one part at a time. There are two hoists over the line for moving parts between stations. The two hoists are on the same track and therefore cannot travel past each other. We denote the hoist on the left as H_1 and the other as H_2. The leftmost position that H_1 can reach is w_l and the rightmost position

Figure 1. An example of an electroplating line.



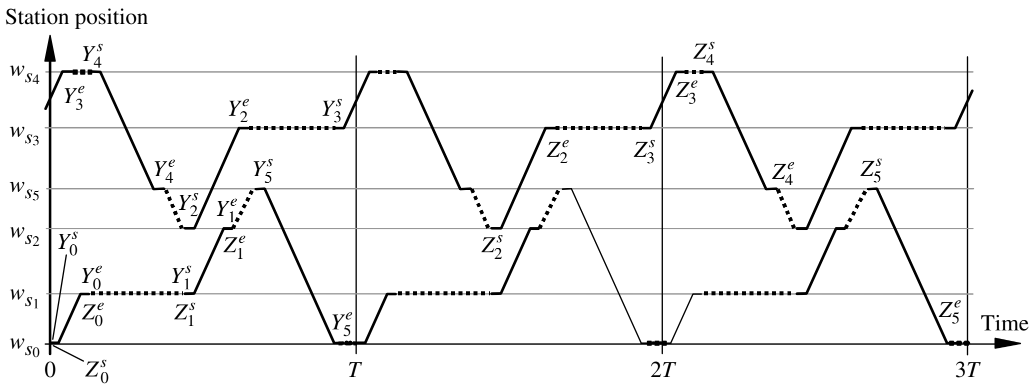
that H_2 can reach is w_r. It is natural to assume that the positions of all stations are within the range between w_l and w_r (because any station outside this range cannot be reached by any hoist). To avoid collision, the two hoists must maintain at least a minimum distance, d, between them.

The electroplating line produces identical parts. The process plan for the parts, i.e., the sequence of stations that each part visits, is given as $s = [s_0, s_1, s_2, \ldots, s_n, s_{n+1}]$, where s_i, $i = 1, 2, \ldots, n$, is the station for the ith processing stage of the part. Station $s_0 = 0$ is the loading station. Station s_{n+1} is the unloading station. For the systems where loading and unloading are at the same station, $s_{n+1} \equiv s_0$. The required processing time at processing station s_i is a given constant, τ_i, $i = 1, 2, \ldots, n$. After processing at station s_i, the part must be immediately moved to station s_{i+1} by a hoist. This move is denoted as m_i. To perform the move, the hoist first lifts up the part at station s_i, then travels to s_{i+1}, and finally lowers the part down and drops it there. The time for the hoist to lower down or rise up is μ. The speed of a hoist carrying a part is ν, and the speed of an empty hoist is λ ($\lambda > \nu$). The total time for m_i can then be expressed as $|w_{s_i} - w_{s_{i+1}}|/\nu + 2\mu$. When an empty hoist moves from one station to another, it can move at a lower position, and therefore the times for lifting up and lowering down are not needed. Therefore, the time for an empty hoist move between s_i and s_j is $|w_{s_i} - w_{s_{i+1}}|/\lambda$.

We consider cyclic production where a cycle is the period between the time points when two adjacent parts enter the system (start the move from s_0 to s_1). Note that after a part completes the processing at station s_1 and is moved away from the station to s_2, another part may be moved into and begin processing at station s_1. Therefore, there can be more than one part being processed at different stations in the system at the same time. It may take more than one cycle time for an individual part to complete the entire process in the system. Figure 2 shows an example of this cyclic production. The horizontal and vertical axes in the diagram represent the time and station positions, respectively. Solid lines indicate the part moves between stations, where a horizontal segment indicates either lifting up or dropping off of a part at a station and an inclined segment indicates the travel between two stations. Dotted inclined lines are empty hoist moves. A dotted horizontal line indicates that a hoist waits at a position. All the loaded and empty moves of a hoist form a path for that hoist. The processing time of a part at a station is not explicitly plotted in the figure, but it can be determined from the loaded moves. It is the duration between the ending point of the move to this station and the starting point of the move away from the same station. The operations in every cycle are exactly the same. A diagram similar to that in Figure 2, but showing one cycle of operations, will be sufficient to demonstrate cyclic production. Such a one-cycle diagram is called a "time-way diagram." We will call any diagram of this type a time-way diagram even when it does not show exactly one cycle.

For the part entering the system at time 0, the starting time and ending time of move m_i for this part can

Figure 2. An example of a time-way diagram.



be computed from the parameters by using the following formulas:

$$Z_i^s = \sum_{k=1}^{i}(2\mu + |w_{s_{k-1}} - w_{s_k}|/\nu + \tau_k),\quad i = 1, \ldots, n, \quad Z_0^s = 0 \quad \text{(starting points)}, \tag{1}$$

$$Z_i^e = Z_i^s + 2\mu + |w_{s_i} - w_{s_{i+1}}|/\nu,\quad i = 0, \ldots, n \quad \text{(ending points)}. \tag{2}$$

In each production cycle, every move (may be for different parts), m_i, $i = 0, 1, \ldots, n$, is performed exactly once. For a given cycle length, T, the starting and ending times of all the required moves in a cycle can be calculated from Z_i^s and Z_i^e as follows:

$$Y_i^s = Z_i^s \bmod T, \quad i = 0, \ldots, n, \tag{3}$$

$$Y_i^e = Z_i^e \bmod T, \quad i = 0, \ldots, n. \tag{4}$$

The relationships among Z_i^s, Z_i^e, Y_i^s, and Y_i^e can be seen in Figure 2. Because every part is introduced to the system at the beginning of a cycle, the difference between Z_i^s (Z_i^e) and Y_i^s (Y_i^e) is an integer multiple of T. Let $\theta_i^s = \lfloor Z_i^s/T \rfloor$ and $\theta_i^e = \lfloor Z_i^e/T \rfloor$. Then, the relationships (3) and (4) can be expressed as

$$Z_i^s = \theta_i^s * T + Y_i^s, \quad i = 0, \ldots, n, \tag{5}$$

$$Z_i^e = \theta_i^e * T + Y_i^e, \quad i = 0, \ldots, n. \tag{6}$$

Note that for different T, the positions of the required moves (Y_i^s and Y_i^e) in the cycle are different. There may or may not exist a feasible hoist schedule to perform all the moves corresponding to a given T.

Definition 1. A cycle length, T, is said to be feasible if there exists a feasible cyclic schedule with this cycle length. A feasible schedule means that there is no more than one part in a tank at any time; each move is assigned to a hoist; between any two part moves assigned to a hoist, there is enough time for the empty move of the hoist; and, at all times, the hoists remain at least the minimum distance, d, apart.

The two-hoist, no-wait cyclic scheduling problem is, then, to find a feasible schedule such that the cycle length, T, is minimized.

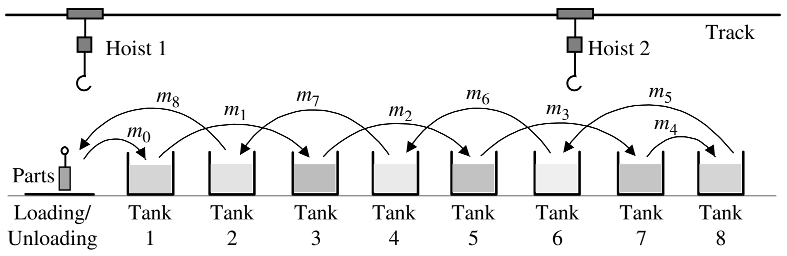
1.2. Literature Review

Most previous research work on the hoist-scheduling problem considers a single-hoist system with processing-time windows and studies one-part cyclic scheduling. This problem was proved to be NP-complete by Lei and Wang (1989). Phillips and Unger (1976), Song et al. (1993), and Liu et al. (2002) developed mixed-integer programming models for the problem. Armstrong et al. (1994), Chen et al. (1998), Lei and Wang (1994), and Ng (1996) applied branch-and-bound algorithms to search for optimal schedules. Two-part cycles were also considered by Lei and Wang (1994). Yin and Yih (1992) proposed a tolerance-based heuristic approach, while Lamothe and Correge (1995) developed a dynamic heuristic for a multiproduct system with random arrivals. A similar problem of scheduling a material-handling robot exists in manufacturing cells without work-in-process buffers. Such manufacturing cells often process several part types simultaneously and the parts are allowed to wait on the machines after processing. Detailed description and classification of the robot-scheduling problems and review of relevant literature can be found in Sriskandarajah et al. (1998) and Crama et al. (2000).

Little work has been done on multihoist systems. Lei and Wang (1991) considered two hoists in a system with a loading station on one end and an unloading station on the other. In this system, parts pass through the system tank by tank in one direction from the loading station to the unloading station. The researchers partitioned all the intertank moves into two nonoverlapping groups and assigned each group to one hoist. The optimal schedule was determined by solving the single-hoist problems alternately. Also by using partitioning, Liu and Zhou (1998) developed a heuristic method to search for the best partitioning point and the corresponding schedule for the two-hoist system, based on simulated annealing. Yang et al. (2001) extended the problem to n hoists and used a simulated annealing algorithm to search for good partition scheduling.

The partitioning policy in these studies is used to avoid the hoist crossing and collision problem. However, one-point clear partitioning may result in only a local optimal solution. Moreover, it can be used only in systems in which the moves can be clearly partitioned. Most systems in practice are not of this type and cannot use such a policy. See Figure 3 for an example.

Figure 3. An electroplating line example on which the partition method does not work.



With certain restrictions, the single-hoist problem may be solved polynomially. Lei (1993) presented a pseudopolynomial algorithm to generate the optimal solution when the sequence of moves is given and the input data are integers. Kats and Levner (1997) developed a polynomial algorithm for the single-robot, one-part cyclic scheduling problem in no-wait flowshop type of manufacturing cells. They found prohibited intervals of the cycle length, T, and obtained the optimal T by searching the allowed intervals. Kats et al. (1999) and Che et al. (2003) extended the method to generate multipart cyclic schedules for the same problem. The single-robot, no-wait scheduling problem is equivalent to the single-hoist, no-wait scheduling problem, although they arise from different applications. For systems with two or more hoists, the no-wait cyclic scheduling problem has not been studied, and whether or not it can be solved polynomially is still an open question.

1.3. Contributions of This Paper

In this paper, we give a positive answer to the above open question. A polynomial algorithm with computational complexity of $O(n^4 \log n)$ is developed to search for an optimal solution.

Our objective is to minimize the cycle length, T. Obviously, if we could check the feasibility of all T values, then the optimal T would be obtained by comparing the feasible ones. However, checking all T values is impossible as T is a continuous variable. On the other hand, we have to guarantee not to miss any feasible solution if we check only a limited number of T values. The prohibited interval idea by Kats and Levner (1997) is helpful to limit the search. We extend this idea here to the two-hoist problem to find some infeasible intervals and avoid checking them. However, as the two-hoist problem has many new features, we derive new T values to further divide the allowed intervals. The feasibility checking problem at each T is also different and involves new decisions. Our polynomial solution to the problem includes the following major developments.

(1) We show that the feasibility property of the problem may only change at a limited number of T values in the T domain. While it is difficult to identify these special values, we find a larger set of T values that includes all these special values. The T values in this larger set will be called "thresholds." At each threshold, some problem properties change, although the feasibility property may or may not change.

(2) We find necessary conditions for hoist assignment and sufficient conditions of infeasibility for any given T. Based on these conditions, we develop an efficient method to check the feasibility of a threshold and, in the feasible case, to construct a feasible hoist schedule.

Solving the two-hoist problem is a significant step towards the possible solution of more general multihoist problems. The two-hoist problem is much more complicated than the single-hoist problem. It involves many decisions and features that do not exist in the single-hoist problem, but that are common for multihoist problems. First, in the two-hoist problem, we need to decide which hoist should be assigned to perform a part move. There is no such decision in the single-hoist case, as all moves must be performed by the only hoist. Second, when more than one hoist runs on the same track, hoist crossings and collisions must be avoided. In addition, with the possibility of hoist collisions, the time for lifting up and dropping off the parts by the hoists must be considered separately from the time for traveling. This is because the hoist must stay in the same position during the lift-up or drop-off time. If these times were included in the traveling time, then the hoist would appear to be moving during the lift-up and drop-off times, and this might make a collision situation appear not to be colliding, or the other way around. With the lift-up and drop-off times considered separately, the analysis becomes even more complicated.

Unlike previous two-hoist studies (Lei and Wang 1991, Liu and Zhou 1998), we allow any part-flow patterns in the system, not necessarily always in one direction from one end to the other. In addition, we do not require a clear, one-point (station) partitioning of the line into two separate one-hoist segments. Rather, we allow overlapping in the working zones of the two hoists as long as they do not collide with each other at any time. Material-handling devices, such as automated guided vehicles, cranes, and hoists running on one track are commonly used in manufacturing and logistics systems. Effectively scheduling them to perform required tasks and to avoid collision at the same time is critical in all these systems. Some ideas in our solution to the two-hoist problem can be useful for this class of problems.

The algorithm developed in this paper applies only to the no-wait problem. However, the no-wait solution is important both in practice and for further research. Although the no-wait constraint may not be necessary in many situations such as robot scheduling in manufacturing cells, it is practical for hoist scheduling in some electroplating systems. In electroplating, parts are processed in chemical liquids in the tanks. Waiting in the liquid means additional processing. If product quality requires fixed processing times, then any waiting will result in defectives. In situations where processing times are allowed to vary in given time windows, the no-wait constraint is relaxed. The problem becomes NP-hard because its special case, the single-hoist problem, was proved NP-hard. Then, any efficient solution will likely be of the heuristic type. In this case, heuristic methods may be developed to search for the best time parameters within their windows, while the no-wait algorithm will be useful in providing an efficient solution to each subproblem with given time parameters.

The rest of this paper is organized as follows. In §2, we first give a lower bound, T_0, and an upper bound, T^0, of the optimal cycle length. The intervals of T that are infeasible due to move conflicts are then identified and taken out from the range $[T_0, T^0]$. A method is presented in §3 to check feasibility of any remaining T and, in case it is feasible, to generate the corresponding hoist schedule. In §4, more thresholds are identified so that the feasibility property only changes at some of the thresholds. The complete algorithm integrating all the results is then presented. Section 5 concludes the paper.

2. Move Conflicts and Thresholds of T Values

The basic idea of our approach to the problem is to identify some thresholds, which are special values of the cycle length T, such that the feasibility property may only change on some of the thresholds, and then to check the feasibility of only these thresholds. If the feasibility of the intervals on both sides of a threshold is the same, then the feasibility of the threshold is also the same as that of the intervals; if the interval on one side is feasible and the interval on the other side is infeasible, we consider the threshold as a feasible point.

To provide limits for searching the optimal cycle length, we first give a lower bound and an upper bound.

Proposition 1. *If there is a feasible solution for the problem, then* $T_0 \equiv \max\{\tau_i \mid i = 1, 2, \ldots, n\}$ *is a lower bound and* $T^0 \equiv Z_n^e + |w_{s_{n+1}} - w_0|/\lambda$ *is an upper bound for the optimal cycle length.*

Proof. Because each station can process at most one part at a time, a feasible cycle length, T, cannot be shorter than the processing time of any station. Therefore, the maximum of the processing times, T_0, is a lower bound of T.

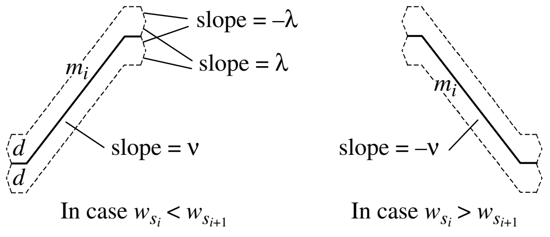
To prove that T^0 is an upper bound, it will be enough to show that it is a feasible cycle length. With the cycle length T^0, it is obvious that a feasible schedule can be obtained as follows: Assign to Hoist 1 all the moves, m_i, with $s_i = 0$ or $s_{i+1} = 0$; assign to Hoist 2 all the moves, m_i, with $s_i = n$ or $s_{i+1} = n$; and assign the rest of the moves to either Hoist 1 or Hoist 2. □

Within the bounds, some T are infeasible and others are feasible. For a specific T, if any two moves are too close to each other in the corresponding time-way diagram, then there will be no way for the two hoists to perform them, and this T will be infeasible. In this case we say that the two moves "conflict" with each other. A formal definition of conflicts between moves is given below.

Definition 2. Two moves are said to conflict with each other if they cannot be performed either by the same hoist due to insufficient time for the empty hoist to travel between them, or by different hoists because, to perform the moves, the two hoists would have to be closer together than the safe distance, d, at some time point.

To illustrate conflicts between two moves, we can imagine a protecting "shell" for one of them in the time-way diagram. The formation of the shell is determined by the safe distance, d, the hoist speed for loaded moves, ν, and the empty hoist speed, λ. Figure 4 shows the shell of a move m_i using dotted lines. Within the period between the

Figure 4. Protecting "shell" of a move.


slope = –λ
slope = λ
m_i
slope = ν
d
d
In case $w_{s_i} < w_{s_{i+1}}$
m_i
slope = –ν
In case $w_{s_i} > w_{s_{i+1}}$

start time and the end time of the move, the dotted lines below and above it are parallel with the move and they vertically remain exactly the safe distance, d, from the move at any point in the period. At two ends of the shell, the inclined short dotted lines connect to the three parallel short horizontal lines exactly at the start or the end time point of the move. The slopes of these short dotted lines are determined by the empty hoist speed, λ.

If any point of a move is in the protecting shell of another move, then the two moves conflict with each other. Now we identify the thresholds that can help to determine, and therefore to eliminate, infeasible intervals due to move conflicts.

Definition 3. For a cycle length, T, if there exists a conflict between any two loaded moves, then this T is said to be Type-1 infeasible.

Consider any two moves, m_i and m_j $(j > i)$, in a cycle. When T increases, the position of m_i, with respect to m_j of a previous part, will move to the right in the time-way diagram. At certain T values, T_{ij}^l, m_j touches the shell of m_i, as shown in Figure 5(i). Increasing T, from T_{ij}^l, will cause the two moves to conflict with each other, as shown in Figure 5(ii), until T reaches another special value, T_{ij}^u, as shown in Figure 5(iii). In this way, the two special T values define a Type-1 infeasible interval of T, (T_{ij}^l, T_{ij}^u).

When $T = T_{ij}^l$, m_j touches the shell of m_i, as in Figure 5(i). If m_j is for the part that is introduced to the system at time 0, then m_i is for the part that is introduced to the system at kT_{ijk}^l (k cycles after the part corresponding to m_j is introduced) for some integer, k. Then, the time at which m_j touches the shell of m_i can be calculated from

Figure 5. Situations when T is at the boundaries of and within a Type-1 infeasible interval.

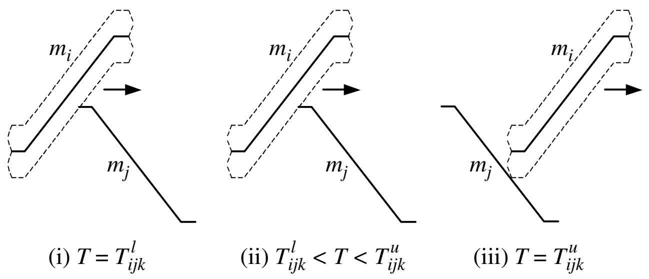
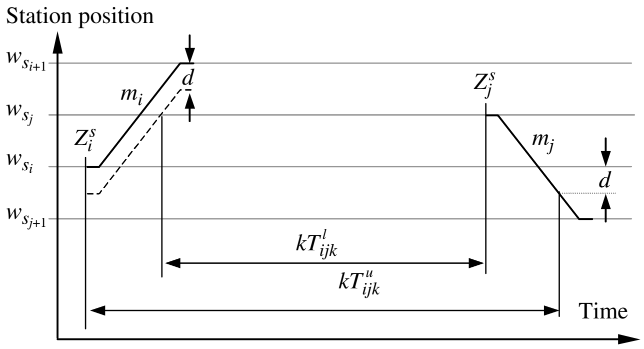


Figure 6. Relationships between Z_i^s, Z_j^s, T_{ijk}^l, and T_{ijk}^u.



both moves, as shown on the two sides of the equation below:

$$Z_j^s = kT_{ijk}^l + Z_i^e - \mu - (|w_{s_j} - w_{s_{i+1}}| - d)/\nu.$$

From this equation, we find

$$T_{ijk}^l = [Z_j^s - Z_i^e + \mu + (|w_{s_j} - w_{s_{i+1}}| - d)/\nu]/k.$$

Similarly, from Figure 5(iii), we find

$$T_{ijk}^u = [Z_j^e - Z_i^s - \mu - (|w_{s_{j+1}} - w_{s_i}| - d)/\nu]/k.$$

These relationships are illustrated in Figure 6.

Note that if $k > j - i$ in the above expressions, the T values would be smaller than T_0 and therefore infeasible. Thus, k can only take values of $1, \ldots, j-i$.

For each $k = 1, \ldots, j-i$, there is a Type-1 infeasible interval (T_{ijk}^l, T_{ijk}^u) with T_{ijk}^l and T_{ijk}^u defined by the above expressions.

Depending on the positions of the two moves, m_j may touch the shell of move m_i in different ways at the boundary of the Type-1 infeasible interval, and the formulas for calculating T_{ijk}^l and T_{ijk}^u can also be different for different cases. Figures 7 and 8 show all the possible cases at T_{ijk}^l and T_{ijk}^u, respectively. Note that there is not a one-to-one correspondence between the cases in the two figures.

Figure 7. Different cases when $T = T_{ijk}^l$.

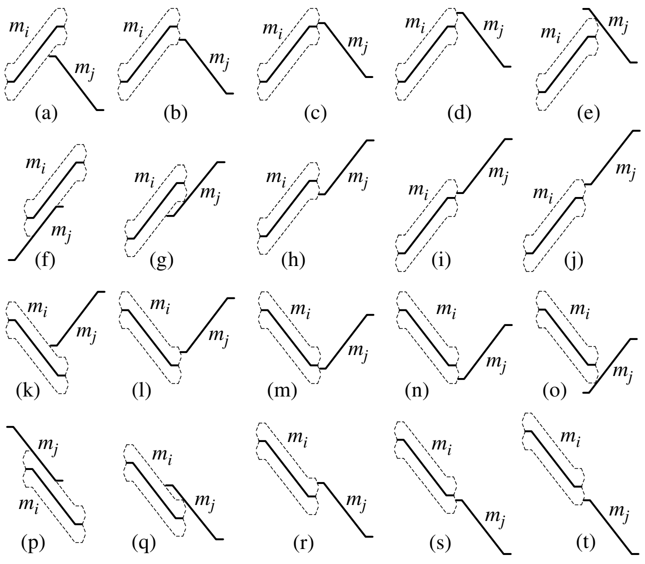
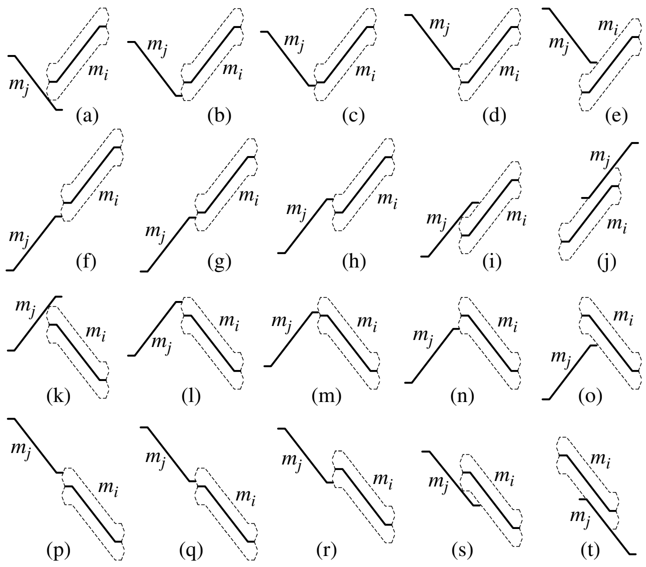


Figure 8. Different cases when $T = T_{ijk}^u$.



All these cases can be described using the relationships among the tank positions of the two moves. Each case at T_{ijk}^l (see Figure 7) can be mathematically expressed using a logical combination of some conditions in the sets A, B, C, and D listed below. Similarly, the mathematical expressions for each of the cases at T_{ijk}^u (see Figure 8) can be written using a logical combination of some conditions in the sets A, B, E, and F below.

$$A = \{w_{s_i} < w_{s_{i+1}}, w_{s_i} > w_{s_{i+1}}\};$$

$$B = \{w_{s_j} < w_{s_{j+1}}, w_{s_j} > w_{s_{j+1}}\};$$

$$\begin{aligned} C = \{&w_{s_j} \leqslant w_{s_i} - d, w_{s_i} - d < w_{s_j} \leqslant w_{s_{i+1}} - d,\\ &w_{s_{i+1}} - d < w_{s_j} < w_{s_{i+1}} - d/2,\\ &w_{s_{i+1}} - d/2 \leqslant w_{s_j} \leqslant w_{s_{i+1}} + d/2,\\ &w_{s_{i+1}} + d/2 < w_{s_j} < w_{s_{i+1}} + d, w_{s_j} \geqslant w_{s_{i+1}} + d,\\ &w_{s_j} \geqslant w_{s_i} + d, w_{s_i} + d > w_{s_j} \geqslant w_{s_{i+1}} + d, w_{s_j} \leqslant w_{s_{i+1}} - d\}; \end{aligned}$$

$$\begin{aligned} D = \{&w_{s_{j+1}} \leqslant w_{s_i} - d, w_{s_i} - d < w_{s_{j+1}} \leqslant w_{s_{i+1}} - d,\\ &w_{s_{j+1}} > w_{s_{i+1}} - d, w_{s_{j+1}} \leqslant w_{s_{i+1}} - d, w_{s_{j+1}} \geqslant w_{s_i} + d,\\ &w_{s_i} + d > w_{s_{j+1}} \geqslant w_{s_{i+1}} + d,\\ &w_{s_{j+1}} < w_{s_{i+1}} + d, w_{s_{j+1}} \geqslant w_{s_{i+1}} + d\}; \end{aligned}$$

$$\begin{aligned} E = \{&w_{s_{j+1}} \leqslant w_{s_{i+1}} - d, w_{s_{i+1}} - d < w_{s_{j+1}} \leqslant w_{s_i} - d,\\ &w_{s_i} - d < w_{s_{j+1}} < w_{s_i} - d/2,\\ &w_{s_i} - d/2 \leqslant w_{s_{j+1}} \leqslant w_{s_i} + d/2,\\ &w_{s_i} + d/2 < w_{s_{j+1}} < w_{s_i} + d, w_{s_{j+1}} \geqslant w_{s_i} + d,\\ &w_{s_{j+1}} \geqslant w_{s_{i+1}} + d, w_{s_{i+1}} + d > w_{s_{j+1}} \geqslant w_{s_i} + d,\\ &w_{s_{j+1}} \leqslant w_{s_i} - d\}; \end{aligned}$$

$$F=\{w_{s_j}\leqslant w_{s_{i+1}}-d,w_{s_{i+1}}-d<w_{s_j}\leqslant w_{s_i}-d,$$
$$w_{s_j}>w_{s_i}-d,w_{s_j}\leqslant w_{s_i}-d,w_{s_j}\geqslant w_{s_{i+1}}+d,$$
$$w_{s_{i+1}}+d>w_{s_j}\geqslant w_{s_i}+d,w_{s_j}<w_{s_i}+d,w_{s_j}\geqslant w_{s_i}+d\}.$$

Based on different cases, the values of T^l_{ijk} and T^u_{ijk} can be calculated as follows:

$$T^l_{ijk}=\begin{cases}[Z^s_j-Z^e_i+\mu+(|w_{s_j}-w_{s_{i+1}}|-d)/\nu]/k & \text{in cases (a, e, g, k, o, q)},\\ [Z^s_j-Z^e_i-(d-|w_{s_j}-w_{s_{i+1}}|)/\lambda]/k & \text{in cases (b, d, h, j, l, n, r, t)},\\ [Z^s_j-Z^e_i-|w_{s_j}-w_{s_{i+1}}|/\lambda]/k & \text{in cases (c, i, m, s)},\\ [Z^s_j-Z^e_i+2\mu+(|w_{s_j}-w_{s_{i+1}}|-d)/\nu]/k & \text{in cases (f, p).}\end{cases} \quad (7)$$

$$T^u_{ijk}=\begin{cases}[Z^e_j-Z^s_i-\mu-(|w_{s_{j+1}}-w_{s_i}|-d)/\nu]/k & \text{in cases (a, e, i, k, o, s)},\\ [Z^e_j-Z^s_i+(d-|w_{s_{j+1}}-w_{s_i}|)/\lambda]/k & \text{in cases (b, d, f, h, l, n, p, r)},\\ [Z^e_j-Z^s_i+|w_{s_{j+1}}-w_{s_i}|/\lambda]/k & \text{in cases (c, g, m, q)},\\ [Z^e_j-Z^s_i-2\mu-(|w_{s_{j+1}}-w_{s_i}|-d)/\nu]/k & \text{in cases (j, t).}\end{cases} \quad (8)$$

Note that if two moves are spatially far away, they will never conflict, and therefore they will not cause a Type-1 infeasible interval. This case is not plotted in Figures 7 and 8. This case can be expressed as

$$\text{Case (u): } \max(w_{s_i},w_{s_{i+1}})+d\leqslant\min(w_{s_j},w_{s_{j+1}})$$
$$\text{or } \max(w_{s_j},w_{s_{j+1}})+d\leqslant\min(w_{s_i},w_{s_{i+1}}).$$

To unify the descriptions for all cases, we define $T^l_{ijk}=T^u_{ijk}=0$ for case (u). Obviously, this will not affect anything in the range $[T_0, T^0]$.

By considering all pairs of m_i and m_j that may conflict, all possible Type-1 infeasible intervals of T can be expressed as

$$(T^l_{ijk},T^u_{ijk}),\quad i=0,1,\ldots,n-1,\ j=i+1,\ldots,n,\ k=1,\ldots,j-i, \quad (9)$$

where T^l_{ijk} and T^u_{ijk} are calculated above.

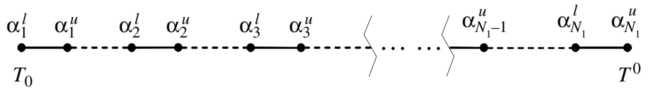
Based on the above analysis, all the T values in $C\equiv\bigcup\{(T^l_{ijk},T^u_{ijk}), i=0,1,\ldots,n-1, j=i+1,\ldots,n, k=1,\ldots,j-i\}$ are infeasible and do not need further investigation. When T is in the set $[T_0,T^0]\backslash C$, there is no Type-1 infeasibility. This set consists of a collection of intervals separated by Type-1 infeasible intervals.

Definition 4. An interval, $[\alpha^l,\alpha^u]$, of cycle length values is called a Type-1 infeasibility-free interval, or simply a Type-1 free interval, if it satisfies the following conditions:

(a) No T in the interval is Type-1 infeasible; and

(b) For any given positive number ϵ, there exists a positive number $\delta<\epsilon$, such that $\alpha^l-\delta$ and $\alpha^u+\delta$ are Type-1 infeasible.

Figure 9. First-layer intervals and thresholds.



The range $[T_0,T^0]$ is then composed of Type-1 infeasible intervals and Type-1 free intervals appearing alternately, as shown in Figure 9. In this figure, the dotted segments are (Type-1) infeasible intervals and the solid segments are Type-1 free intervals.

The feasibility properties of different points in the same Type-1 free interval may be different. We will later find more thresholds in each Type-1 free interval. Therefore, we will refer to the boundaries of Type-1 free intervals as *first-layer thresholds*.

Note that the conflict intervals expressed in (9) may be overlapping. Only some of the T^l_{ijk} and T^u_{ijk} values are thresholds separating the infeasible and Type-1 free intervals in $[T_0,T^0]$. Procedure 1 below generates all the Type-1 free intervals.

Procedure 1. Obtain Type-1 Free Intervals and First-Layer Thresholds

Step 1. Calculate the bounds of Type-1 infeasible intervals (T^l_{ijk},T^u_{ijk}), $i=0,\ldots,n-1$; $j=i+1,\ldots,n$, $k=1,\ldots,j-i$.

Step 2. Sequence the Type-1 infeasible intervals in ascending order of T^l_{ijk}.

Step 3. Remove all these Type-1 infeasible intervals from the range $[T_0,T^0]$ to obtain the Type-1 free intervals (α^l_p,α^u_p), $p=1,\ldots,N_1$. Stop.

Proposition 2. *Procedure* 1 *identifies all Type-*1 *free intervals,* (α^l_p,α^u_p), $p=1,2,\ldots,N_1$. *The computational complexity of the procedure is* $O(n^3\log n)$.

Proof. Expression (9) gives all Type-1 infeasible intervals. Procedure 1 generates all these intervals, sequencing them, and then removes all of them from $[T_0,T^0]$. Therefore, the remaining intervals identified in the procedure are all Type-1 free intervals.

From the ranges of i,j,k in Step 1 of the procedure, it can be seen that the complexity for generating all the Type-1 infeasible intervals is $O(n^3)$. The complexity for sequencing them is $O(n^3\log n)$. Removal of these intervals from $[T_0,T^0]$ can be done in $O(n^3)$ time. Because these three parts are in series, the complexity of the whole procedure is $O(n^3\log n)$. □

3. Hoist Scheduling for a Given Type-1 Free T

In this section, we present a method for constructing a feasible schedule or identifying infeasibility for a given Type-1 free T through necessary assignment of moves to hoists. Given any Type-1 free T, we can calculate the corresponding values of Z_i^s, Z_i^e, Y_i^s, and Y_i^e, $i = 0, 1, \ldots, n$. To construct a feasible schedule, we need to assign each of the moves to a hoist and to find a feasible path for each hoist so that the two paths have at least a distance of d at any time in the cycle. Feasibility also requires that the end of the path of a hoist in a cycle connects to the beginning of the path of the hoist in the next cycle. Therefore, when we connect the moves assigned to a hoist to form a path in a cycle, we need to connect the last move to the first move assigned to the hoist in the next cycle. For convenience, in the following discussion, we define

$$Y_i^f = \begin{cases} Y_i^e & \text{if } Y_i^e > Y_i^s \\ Y_i^e + T & \text{if } Y_i^e < Y_i^s, \end{cases} \qquad i = 0, 1, \ldots, n.$$

We sequence the moves in the order of their start times, Y_i^s, $i = 0, 1, \ldots, n$, and denote the starting and ending times of the ith move in the sequence as $Y_{[i]}^s$ and $Y_{[i]}^f$, $i = 0, 1, \ldots, n$, respectively.

We further define $m_{[0]}$ to $m_{[n]}$ of the next cycle as $m_{[n+1]}$ to $m_{[2n+1]}$ for the cycle under consideration. Then, we have

$$Y_{[i+n+1]}^s = T + Y_{[i]}^s, \quad Y_{[i+n+1]}^f = T + Y_{[i]}^f, \quad i = 0, \ldots, n.$$

3.1. Necessary Conditions for Hoist Assignments

We first discuss the necessary conditions for hoist assignment concerning a single move. If the position of a tank is closer than d to one end of the electroplating line, then this tank can be reached by only one hoist. This is because even if this hoist goes to the end of the line, the other hoist cannot reach the tank due to the safety distance between the two hoists. Therefore, if a move is from or to such a tank, the move has to be assigned to one specific hoist. These necessary hoist assignments can be expressed as follows:

Assign $m_{[i]}$ to H_1 if $\min(w_{s_{[i]}}, w_{s_{[i]+1}}) < w_l + d$,

Assign $m_{[i]}$ to H_2 if $\max(w_{s_{[i]}}, w_{s_{[i]+1}}) > w_r - d$.

Now, we discuss the necessary hoist assignments caused by the relative positions of two moves in the time-way diagram. Consider a pair of moves, $m_{[i]}$ and $m_{[j]}$, $i < j$. It is clear that there are only four possible combinations of assigning them to the two hoists as listed below:

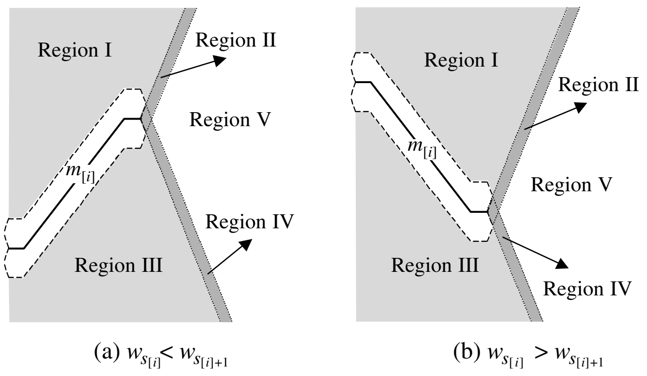
Combination (1). $m_{[i]}$ and $m_{[j]}$ are both assigned to H_1.

Combination (2). $m_{[i]}$ and $m_{[j]}$ are both assigned to H_2.

Combination (3). $m_{[i]}$ is assigned to H_1 and $m_{[j]}$ is assigned to H_2.

Combination (4). $m_{[i]}$ is assigned to H_2 and $m_{[j]}$ is assigned to H_1.

Figure 10. The regions for the starting point of $m_{[j]}$ with respect to $m_{[i]}$.



The hoist assignments for the pair of moves depend on their relative positions in the time-way diagram. For any Type-1 free T, $m_{[j]}$ must be outside the shell of $m_{[i]}$. Figure 10 shows $m_{[i]}$ and all possible regions for the starting point of $m_{[j]}$. The line separating Regions I and II belongs to Region II, and its slope is λ. The line separating Regions II and V belongs to Region V, and its slope is also λ. Similarly, the slope of the lines separating Regions III, IV, and V is $-\lambda$, and each line belongs to the region on its right. Note that there is a small diamond-shaped area belonging to both Regions II and IV. This does not affect the analysis.

For convenient mathematical expression, we define the following function:

$$f_k(t) = \begin{cases} w_{s_k}, & Y_k^s \leqslant t \leqslant Y_k^s + \mu, \\ w_{s_k} + \nu(t - Y_k^s - \mu), & \\ \qquad Y_k^s + \mu < t \leqslant Y_k^f - \mu \text{ and if } w_{s_k} < w_{s_{k+1}}, & \\ w_{s_k} - \nu(t - Y_k^s - \mu), & \\ \qquad Y_k^s + \mu < t \leqslant Y_k^f - \mu \text{ and if } w_{s_k} > w_{s_{k+1}}, & \\ w_{s_{k+1}}, & t > Y_k^f - \mu. \end{cases}$$

We can see that this function represents the line segments of move k in the time-way diagram with the last segment extended to the right. Then, the expression $w_{s_{[j]}} > f_{[i]}(Y_{[j]}^s)$ and $\lambda(Y_{[j]}^s - Y_{[i]}^f) < w_{s_{[j]}} - w_{s_{[i]+1}}$ indicates that the starting point of $m_{[j]}$ is in the area that includes Region I and the upper half of the shell of $m_{[i]}$ (see Figure 10). For any Type-1 free T, because the starting point of $m_{[j]}$ will never be in the shell of $m_{[i]}$, the expression is practically equivalent to the starting point of $m_{[j]}$ being in Region I of $m_{[i]}$. Similarly, we can obtain the expressions for other regions as listed below. For conciseness, we will simply use $m_{[j]} \in R_{\mathrm{I}}(m_{[i]})$ to mean that the starting point of $m_{[j]}$ is in Region I of $m_{[i]}$. Similar notation will be used for other regions:

$$m_{[j]} \in R_{\mathrm{I}}(m_{[i]})\colon\ w_{s_{[j]}} > f_{[i]}(Y_{[j]}^s) \quad \text{and} \quad \lambda(Y_{[j]}^s - Y_{[i]}^f) < w_{s_{[j]}} - w_{s_{[i]+1}},$$

$$m_{[j]} \in R_{\mathrm{II}}(m_{[i]})\colon\ w_{s_{[j]}} - w_{s_{[i]+1}} \leqslant \lambda(Y_{[j]}^s - Y_{[i]}^f) < w_{s_{[j]}} - w_{s_{[i]+1}} + d \quad \text{and } Y_{[j]}^s - Y_{[i]}^f \geqslant 0,$$

$$m_{[j]} \in R_{\mathrm{III}}(m_{[i]})\colon\ w_{s_{[j]}} < f_{[i]}(Y^s_{[j]}) \quad \text{and} \quad \lambda(Y^s_{[j]} - Y^f_{[i]}) < w_{s_{[i]+1}} - w_{s_{[j]}},$$

$$m_{[j]} \in R_{\mathrm{IV}}(m_{[i]})\colon\ w_{s_{[i]+1}} - w_{s_{[j]}} \leqslant \lambda(Y^s_{[j]} - Y^f_{[i]}) < w_{s_{[i]+1}} + d - w_{s_{[j]}} \quad \text{and } Y^s_{[j]} - Y^f_{[i]} \geqslant 0,$$

$$m_{[j]} \in R_{\mathrm{V}}(m_{[i]})\colon\ \lambda(Y^s_{[j]} - Y^f_{[i]}) \geqslant |w_{s_{[i]+1}} - w_{s_{[j]}}| + d.$$

Now we analyze the situation of hoist assignments related to each region.

$m_{[j]} \in R_{\mathrm{I}}(m_{[i]})$. In this case, hoist assignment combinations (1) and (2) are infeasible because a hoist will not have enough time to travel to the starting point of $m_{[j]}$ after finishing $m_{[i]}$. Assignment combination (4) is also infeasible because, if $m_{[i]}$ is assigned to H_2, H_1 (always below H_2) cannot reach Region I. As a result, assignment combination (3) is necessary for feasibility.

$m_{[j]} \in R_{\mathrm{III}}(m_{[i]})$. Similarly, assignment combination (4) is necessary for this case.

$m_{[j]} \in R_{\mathrm{II}}(m_{[i]})$. In this case, if $m_{[i]}$ is assigned to H_2, then H_1 must be below the shell of $m_{[i]}$ and there will not be enough time for H_1 to reach the starting point of $m_{[j]}$. Therefore, $m_{[j]}$ must also be assigned to H_2. Conversely, if $m_{[j]}$ is assigned to H_1, $m_{[i]}$ must also be assigned to H_1.

$m_{[j]} \in R_{\mathrm{IV}}(m_{[i]})$. Similarly, in this case, if $m_{[i]}$ is assigned to H_1, $m_{[j]}$ must also be assigned to H_1; if $m_{[j]}$ is assigned to H_2, $m_{[i]}$ must also be assigned to H_2.

$m_{[j]} \in R_{\mathrm{V}}(m_{[i]})$. For this case, the two moves are far away from each other in the time dimension and any assignment combination can be feasible.

Based on the above analysis, we can now summarize the necessary conditions for hoist assignments in different situations.

Proposition 3. *In any feasible schedule with cycle length T, the following move-to-hoist assignments are necessary*:

(1) *For any move m_i, if $\min\{w_{s_i}, w_{s_{i+1}}\} < w_l + d$, then m_i must be assigned to H_1.*

(2) *For any move m_i, if $\max\{w_{s_i}, w_{s_{i+1}}\} > w_r - d$, then m_i must be assigned to H_2.*

(3) *For any pair of moves, $m_{[i]}$ and $m_{[j]}$ $(i < j)$, if $m_{[j]} \in R_{\mathrm{I}}(m_{[i]})$, then $m_{[i]}$ and $m_{[j]}$ must be assigned to H_1 and H_2, respectively.*

(4) *For any pair of moves, $m_{[i]}$ and $m_{[j]}$ $(i < j)$, if $m_{[j]} \in R_{\mathrm{III}}(m_{[i]})$, then $m_{[i]}$ and $m_{[j]}$ must be assigned to H_2 and H_1, respectively.*

(5) *For any pair of moves, $m_{[i]}$ and $m_{[j]}$ $(i < j)$, if $m_{[j]} \in R_{\mathrm{IV}}(m_{[i]})$ and if $m_{[i]}$ has been assigned to H_1, then $m_{[j]}$ must also be assigned to H_1.*

(6) *For any pair of moves, $m_{[i]}$ and $m_{[j]}$ $(i < j)$, if $m_{[j]} \in R_{\mathrm{II}}(m_{[i]})$ and if $m_{[i]}$ has been assigned to H_2, then $m_{[j]}$ must also be assigned to H_2.*

(7) *For any pair of moves, $m_{[i]}$ and $m_{[j]}$ $(i < j)$, if $m_{[j]} \in R_{\mathrm{IV}}(m_{[i]})$ and if $m_{[j]}$ has been assigned to H_2, then $m_{[i]}$ must also be assigned to H_2.*

(8) *For any pair of moves, $m_{[i]}$ and $m_{[j]}$ $(i < j)$, if $m_{[j]} \in R_{\mathrm{II}}(m_{[i]})$ and if $m_{[j]}$ has been assigned to H_1, then $m_{[i]}$ must also be assigned to H_1.*

Proof. In the situation of Item (1) or (2), part of the move is in the spatial area that only one hoist can reach; therefore, the assignment of the move to that hoist is necessary.

Items (3) and (4) make assignments for situations where $m_{[j]} \in R_{\mathrm{I}}(m_{[i]})$ and $m_{[j]} \in R_{\mathrm{III}}(m_{[i]})$, respectively. Items (5) and (7) make assignments for situations where $m_{[j]} \in R_{\mathrm{IV}}(m_{[i]})$. Items (6) and (8) make assignments for the situations where $m_{[j]} \in R_{\mathrm{II}}(m_{[i]})$. All these assignments have been shown to be necessary in the earlier analysis. □

3.2. Feasibility Checking

According to the conditions in Proposition 3, we can make all necessary assignments of moves to the hoists until no more assignments can be made. Then, we can check whether the assignments can form a feasible schedule. From here onwards we will refer to Item (1) in Proposition 3 as P3(1) for short. Other items will be referred to in a similar way.

Proposition 4. *For a given Type-1 free cycle length T, after all the necessary hoist assignments according to Proposition 3, if any move has been assigned to both hoists, then T is infeasible. Otherwise, a feasible schedule can be constructed with cycle length T.*

Proof. It is obvious that one move cannot be performed by two hoists. Thus, T is infeasible if it is necessary to assign any one move to both hoists.

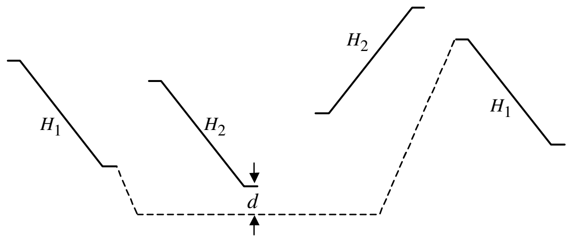
If no move is assigned to both hoists after all the necessary assignments, we can construct a feasible schedule in the following two steps.

Assign free moves (those still unassigned). The free moves can be assigned, all together, to the same hoist and it can be either of the two hoists. Suppose all the free moves are assigned to H_2. Now every move is uniquely assigned to a hoist.

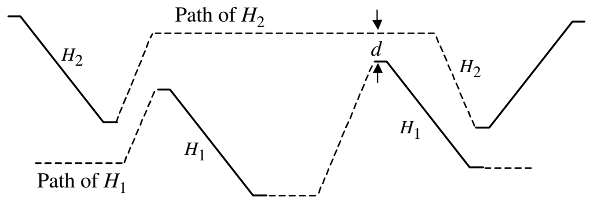
Construct a feasible path for each hoist. First, list the moves assigned to the same hoist in the ascending order of their starting times and let F_1 and F_2 be the ordered lists of the moves assigned to H_1 and H_2, respectively.

For any two moves, $m_{[i1]}$ and $m_{[i2]}$, which are adjacent in F_1, let $S2$ be the set of moves in F_2 whose starting times are between $Y^f_{[i1]}$ and $Y^s_{[i2]}$ and let $E2$ be the set of moves in F_2 whose ending times are between $Y^f_{[i1]}$ and $Y^s_{[i2]}$. Draw a horizontal line in the time-way diagram passing the position $w = \min\{w_{s_{[i1]+1}}, w_{s_{[i2]}}, w_{s_j} - d, w_{s_{k+1}} - d \mid j \in S2, k \in E2\}$. From the end point of $m_{[i1]}$, draw a line downwards with slope $-\lambda$; from the start point of $m_{[i2]}$, draw a line downwards with slope λ until the lines meet the horizontal line. The three line segments form a path for H_1 between $m_{[i1]}$ and $m_{[i2]}$ (see Figure 11a for an example). Note that some segments of the path may be of zero length.

Figure 11. Feasible path for a hoist between two moves assigned to it.



(a) A three-segment path between two adjacent moves in F_1



(b) A three-segment path between two adjacent moves in F_2

Because T is Type-1 free, the moves in F_2 are vertically at least d distance above $m_{[i1]}$ and $m_{[i2]}$ in the time-way diagram. For any $m_j \in F_2$, feasibility requires that $m_j \notin F_1$. According to Proposition 3, we have $m_j \notin R_{\mathrm{III}} \cup R_{\mathrm{IV}}(m_{[i1]})$ and $m_{[i2]} \notin R_I \cup R_{\mathrm{II}}(m_j)$. Therefore, all moves that are in F_2 and between $m_{[i1]}$ and $m_{[i2]}$ will be at least d distance above the two sloped segments on the path of H_1. These moves are also at least d distance above the horizontal segment of the H_1 path according to the definition of the horizontal line. Therefore, the path of H_1 will be at least d distance from the moves of H_2. Linking every adjacent two moves in F_1 in this way, we can get a complete path for H_1 that maintains at least distance d from the moves of H_2 at any time.

Similarly, for any two moves, $m_{[i1]}$ and $m_{[i2]}$, which are adjacent in F_2, let $S1$ be the set of moves in F_1 whose starting times are between $Y^f_{[i1]}$ and $Y^s_{[i2]}$, and let $E1$ be the set of moves in F_1 whose ending times are between $Y^f_{[i1]}$ and $Y^s_{[i2]}$. Draw a horizontal line in the time-way diagram passing the position $w = \max\{w_{s_{[i1]+1}}, w_{s_{[i2]}}, w_{s_j} + d, w_{s_{k+1}} + d \mid j \in S1, k \in E1\}$. From the end point of $m_{[i1]}$, draw a line upwards with slope λ; from the start point of $m_{[i2]}$, draw a line upwards with slope $-\lambda$ until the lines meet the horizontal line. The three line segments form a path for H_2 between $m_{[i1]}$ and $m_{[i2]}$ (see Figure 11b for an example). Linking every adjacent two moves in F_2 like this, we can get a complete path for H_2. Between any two moves $m_{[i1]}$ and $m_{[i2]}$ adjacent in F_2, if there is any point on the path of H_1 higher than $\max\{w_{s_{[i1]+1}}, w_{s_{[i2]}}\}$, then the highest point must be a start or an end point of a move in $S1$ or $E1$. From the above construction method and the slopes of the linear segments, we can see that the path of H_2 maintains at least distance d from the path of H_1 at any time. □

3.3. An Efficient Way to Make All Necessary Hoist Assignments

Propositions 3 and 4 provide the basis for developing a procedure to check the feasibility of any Type-1 free T and to construct a feasible schedule in case T is feasible. The critical part of such a procedure is to ensure that all necessary hoist assignments are made before the T is confirmed to be feasible. It is clear that all the assignments due to P3(1) and P3(2) can be made by checking these two conditions for each move once, because these conditions are only related to individual moves. Other assignment conditions are related to a pair of moves. To make all necessary assignments, it would be enough to check the conditions for every pair of moves repeatedly until no assignment can be made for a complete round of checking. Such a method would take much more checking than necessary. We now give some properties of the problem that enable a more efficient way to make all the necessary hoist assignments.

Proposition 5. *For any three moves, $m_{[i]}$, $m_{[j]}$, and $m_{[k]}$, $i < j < k$, in a time-way diagram corresponding to a Type-1 free cycle length, we have:*

(1) *If $m_{[j]} \in R_{\mathrm{II}} \cup R_{\mathrm{IV}} \cup R_{\mathrm{V}}(m_{[i]})$ and $m_{[k]} \in R_{\mathrm{I}}(m_{[i]})$, then $m_{[k]} \in R_{\mathrm{I}}(m_{[j]})$.*

(2) *If $m_{[j]} \in R_{\mathrm{II}} \cup R_{\mathrm{IV}} \cup R_{\mathrm{V}}(m_{[i]})$ and $m_{[k]} \in R_{\mathrm{III}}(m_{[i]})$, then $m_{[k]} \in R_{\mathrm{III}}(m_{[j]})$.*

(3) *If $m_{[j]} \in R_{\mathrm{II}} \cup R_{\mathrm{IV}} \cup R_{\mathrm{V}}(m_{[i]})$ and $m_{[k]} \in R_{\mathrm{II}}(m_{[i]})$, then $m_{[k]} \in R_{\mathrm{I}} \cup R_{\mathrm{II}}(m_{[j]})$.*

(4) *If $m_{[j]} \in R_{\mathrm{II}} \cup R_{\mathrm{IV}} \cup R_{\mathrm{V}}(m_{[i]})$ and $m_{[k]} \in R_{\mathrm{IV}}(m_{[i]})$, then $m_{[k]} \in R_{\mathrm{III}} \cup R_{\mathrm{IV}}(m_{[j]})$.*

(5) *If $m_{[k]} \in R_{\mathrm{II}} \cup R_{\mathrm{IV}} \cup R_{\mathrm{V}}(m_{[j]})$ and $m_{[j]} \in R_{\mathrm{V}} \cup R_{\mathrm{II}} \backslash R_{\mathrm{IV}}(m_{[i]})$, then $m_{[k]} \in R_{\mathrm{V}} \cup R_{\mathrm{II}} \backslash R_{\mathrm{IV}}(m_{[i]})$.*

(6) *If $m_{[k]} \in R_{\mathrm{II}} \cup R_{\mathrm{IV}} \cup R_{\mathrm{V}}(m_{[j]})$ and $m_{[j]} \in R_{\mathrm{V}} \cup R_{\mathrm{IV}} \backslash R_{\mathrm{II}}(m_{[i]})$, then $m_{[k]} \in R_{\mathrm{V}} \cup R_{\mathrm{IV}} \backslash R_{\mathrm{II}}(m_{[i]})$.*

(7) *If $m_{[k]} \in R_{\mathrm{II}} \cup R_{\mathrm{IV}} \cup R_{\mathrm{V}}(m_{[j]})$ and $m_{[j]} \in R_{\mathrm{II}} \cup R_{\mathrm{IV}} \cup R_{\mathrm{V}}(m_{[i]})$, then $m_{[k]} \in R_{\mathrm{II}} \cup R_{\mathrm{IV}} \cup R_{\mathrm{V}}(m_{[i]})$.*

Proof. (1) From Figure 10, we can see that $m_{[j]} \in R_{\mathrm{II}} \cup R_{\mathrm{IV}} \cup R_{\mathrm{V}}(m_{[i]})$ implies $Y^s_{[j]} \geqslant Y^f_{[i]}$ and $w_{s_{[j]}} \leqslant w_{s_{[i]+1}} + \lambda(Y^s_{[j]} - Y^f_{[i]})$; i.e., in the time-way diagram, the starting point of $m_{[j]}$ is on or below the line $w_{s_{[i]+1}} + \lambda(t - Y^f_{[i]})$, which is the line between Regions I and II of $m_{[i]}$ and which has a slope of λ. Because the slopes of all the line segments on $m_{[j]}$ are less than λ, all the points on $m_{[j]}$ are also on or below this line; i.e., $f_{[j]}(t) \leqslant w_{s_{[i]+1}} + \lambda(t - Y^f_{[i]})$ for all $t \geqslant Y^s_{[j]}$. For the end point of $m_{[j]}$, we have $w_{s_{[j]+1}} = f_{[j]}(Y^f_{[j]}) \leqslant w_{s_{[i]+1}} + \lambda(Y^f_{[j]} - Y^f_{[i]})$ and $w_{s_{[j]+1}} + \lambda(t - Y^f_{[j]}) = w_{s_{[j]+1}} - \lambda(Y^f_{[j]} - Y^f_{[i]}) + \lambda(t - Y^f_{[i]}) \leqslant w_{s_{[i]+1}} + \lambda(t - Y^f_{[i]})$ for all $t \geqslant Y^s_{[j]}$. Then, from $m_{[k]} \in R_{\mathrm{I}}(m_{[i]})$, we obtain $w_{s_{[k]}} > w_{s_{[i]+1}} + \lambda(Y^s_{[k]} - Y^f_{[i]}) \geqslant f_{[j]}(Y^s_{[k]})$ and $w_{s_{[k]}} > w_{s_{[i]+1}} + \lambda(Y^s_{[k]} - Y^f_{[i]}) \geqslant w_{s_{[j]+1}} + \lambda(Y^s_{[k]} - Y^f_{[j]})$. Therefore, $m_{[k]} \in R_{\mathrm{I}}(m_{[j]})$.

(2) In the time-way diagram, this situation is a vertical mirror image of (1). The proof can be done in a similar way.

(3) From proof of Item (1), we know that $m_{[j]} \in R_{\mathrm{II}} \cup R_{\mathrm{IV}} \cup R_{\mathrm{V}}(m_{[i]})$ implies $f_{[j]}(t) \leqslant w_{s_{[i]+1}} + \lambda(t - Y^f_{[i]})$ and

$w_{s_{[j]+1}} + \lambda(t - Y^f_{[j]}) \leqslant w_{s_{[i]+1}} + \lambda(t - Y^f_{[i]})$ for all $t \geqslant Y^s_{[j]}$. Then, if $m_{[k]} \in R_{\mathrm{II}}(m_{[i]})$, we have $w_{s_{[k]}} > w_{s_{[i]+1}} + \lambda(Y^s_{[k]} - Y^f_{[i]}) - d \geqslant f_{[j]}(Y^s_{[k]}) - d$ and $w_{s_{[k]}} > w_{s_{[i]+1}} + \lambda(Y^s_{[k]} - Y^f_{[i]}) - d \geqslant w_{s_{[j]+1}} - d + \lambda(Y^s_{[k]} - Y^f_{[j]})$. This indicates that the starting point of $m_{[k]}$ is in Region I or Region II or the shell of $m_{[j]}$. Because the starting point of $m_{[k]}$ will not be in the shell of $m_{[j]}$ for any Type-1 free T, we get $m_{[k]} \in R_{\mathrm{I}} \cup R_{\mathrm{II}}(m_{[j]})$.

(4) The proof is similar to (3).

(5) It can be seen from Figure 10 that $m_{[k]} \in R_{\mathrm{II}} \cup R_{\mathrm{IV}} \cup R_{\mathrm{V}}(m_{[j]})$ implies $Y^s_{[k]} \geqslant Y^f_{[j]}$ and $|w_{s_{[k]}} - w_{s_{[j]+1}}| \leqslant \lambda(Y^s_{[k]} - Y^f_{[j]})$. $m_{[j]} \in R_{\mathrm{V}} \cup R_{\mathrm{II}} \backslash R_{\mathrm{IV}}(m_{[i]})$ implies $Y^s_{[j]} \geqslant Y^f_{[i]} + d/2\lambda$ and $|w_{s_{[j]}} - w_{s_{[i]+1}} - d/2| \leqslant \lambda(Y^s_{[j]} - Y^f_{[i]})$. Because the slopes of all the line segments on $m_{[j]}$ are greater than $-\lambda$ and less than λ, we have $|w_{s_{[j]+1}} - w_{s_{[i]+1}} - d/2| \leqslant \lambda(Y^f_{[j]} - Y^f_{[i]})$. Then, $|w_{s_{[k]}} - w_{s_{[i]+1}} - d/2| = |w_{s_{[k]}} - w_{s_{[j]+1}} + w_{s_{[j]+1}} - w_{s_{[i]+1}} - d/2| \leqslant |w_{s_{[k]}} - w_{s_{[j]+1}}| + |w_{s_{[j]+1}} - w_{s_{[i]+1}} - d/2| \leqslant \lambda(Y^s_{[k]} - Y^f_{[j]} + Y^f_{[j]} - Y^f_{[i]}) = \lambda(Y^s_{[k]} - Y^f_{[i]})$, and $Y^s_{[k]} \geqslant Y^f_{[j]} > Y^s_{[j]} \geqslant Y^f_{[i]} + d/2\lambda$. This indicates $m_{[k]} \in R_{\mathrm{V}} \cup R_{\mathrm{II}} \backslash R_{\mathrm{IV}}(m_{[i]})$.

(6)–(7) The proof is similar to (5). □

Based on this proposition, we have the following recursive relations for making the necessary hoist assignments to the moves.

Proposition 6. (1) *Suppose that all necessary hoist assignments to every move $m_{[i]}$ $(i < j)$, related to move pairs $(m_{[h]}, m_{[i]})$ for all $h < i$, have been made and there is no infeasibility in these assignments. Let F_1 and F_2 be the sets of moves that have been assigned to H_1 and H_2, respectively. Let $i_1 = \max\{i \mid i < j, m_{[i]} \in F_1\}$ and $i_2 = \max\{i \mid i < j, m_{[i]} \in F_2\}$. Then, all necessary hoist assignments to $m_{[j]}$ due to the move pairs $(m_{[h]}, m_{[j]})$ for all $h < j$, can be made by checking only three move pairs $(m_{[i_1]}, m_{[j]})$, $(m_{[i_2]}, m_{[j]})$, and $(m_{[j-1]}, m_{[j]})$.*

(2) *Suppose that all necessary hoist assignment to $m_{[q]}$, caused by move pair $(m_{[p]}, m_{[q]})$, have been made for all p, q, $p < q$; all necessary hoist assignments to every move $m_{[j]}$ $(j > i)$, related to move pairs $(m_{[j]}, m_{[k]})$ for all $k > j$, have been made; and there is no infeasibility in these assignments. Let F_1 and F_2 be the sets of moves that have been assigned to H_1 and H_2, respectively. Let $j_1 = \min\{j \mid j > i, m_{[j]} \in F_1\}$ and $j_2 = \min\{j \mid j > i, m_{[j]} \in F_2\}$. Then, all necessary hoist assignments to $m_{[i]}$ due to the move pairs $(m_{[i]}, m_{[k]})$ for all $k > i$, can be made by checking only two move pairs $(m_{[i]}, m_{[j_1]})$ and $(m_{[i]}, m_{[j_2]})$.*

Proof. (1) For any $h < j - 1$, $m_{[h]}$ can belong to only one of the following three categories: in F_1, in F_2, not in $F_1 \cup F_2$.

If $m_{[h]} \notin F_1 \cup F_2$, then $m_{[j-1]} \in R_{\mathrm{II}} \cup R_{\mathrm{IV}} \cup R_{\mathrm{V}}(m_{[h]})$, because otherwise $m_{[j-1]} \in R_I \cup R_{\mathrm{III}}(m_{[h]})$ and $m_{[h]}$ would have been in F_1 or F_2 based on P3(3) or P3(4). According to Proposition 5(1), if $m_{[j]} \in R_{\mathrm{I}}(m_{[h]})$, then $m_{[j]} \in R_{\mathrm{I}}(m_{[j-1]})$. Checking the conditions in P3, we can see that move pair $(m_{[h]}, m_{[j]})$ will not cause different necessary hoist assignments to $m_{[j]}$ from that caused by pair $(m_{[j-1]}, m_{[j]})$. Similarly, if $m_{[j]}$ is in other regions of $m_{[h]}$, based on P5(2), P5(3), P5(4), and P3, $(m_{[h]}, m_{[j]})$ will not cause different necessary hoist assignments to $m_{[j]}$ from that caused by $(m_{[j-1]}, m_{[j]})$, either.

If $m_{[h]} \in F_1$ and $m_{[h]} \neq m_{[i_1]}$, then $m_{[i_1]} \in R_{\mathrm{II}} \cup R_{\mathrm{IV}} \cup R_{\mathrm{V}}(m_{[h]})$, because otherwise $m_{[h]}$ would have also been in F_2 based on P3(3) or P3(4) indicating infeasibility. Again according to P5(1), P5(2), P5(3), P5(4), and P3, the pair $(m_{[h]}, m_{[j]})$ will not cause different necessary hoist assignments to $m_{[j]}$ from that caused by the pair $(m_{[i_1]}, m_{[j]})$. Similarly, if $m_{[h]} \in F_2$ and $m_{[h]} \neq m_{[i_2]}$, then the pair $(m_{[h]}, m_{[j]})$ will not cause different necessary hoist assignments to $m_{[i]}$ from that caused by the pair $(m_{[i_2]}, m_{[i]})$.

Summarizing all of the above, we can conclude that all necessary hoist assignments to $m_{[j]}$ due to the move pairs $(m_{[h]}, m_{[j]})$ for all $h < i$, can be made by checking only three move pairs $(m_{[i_1]}, m_{[j]})$, $(m_{[i_2]}, m_{[j]})$, and $(m_{[j-1]}, m_{[j]})$.

(2) For any $k > i$, $m_{[k]}$ can belong to only one of the following three categories: in F_1, in F_2, not in $F_1 \cup F_2$.

In case $m_{[k]} \in F_1$ and $m_{[k]} \neq m_{[j_1]}$, $m_{[k]} \notin R_{\mathrm{I}}(m_{[j_1]})$ because otherwise $m_{[k]}$ would have been in F_2 indicating infeasibility. From P4, we know that $(m_{[i]}, m_{[k]})$ can cause hoist assignment to $m_{[i]}$ only if $m_{[k]} \in R_{\mathrm{II}}(m_{[i]})$ or $m_{[k]} \in R_{\mathrm{III}}(m_{[i]})$. Now we consider all possible positions of $m_{[j_1]}$: From $m_{[j_1]} \in F_1$, we know $m_{[j_1]} \notin R_{\mathrm{I}}(m_{[i]})$. If $m_{[j_1]} \in R_{\mathrm{III}}(m_{[i]})$, then $m_{[i]} \in F_2$ and $m_{[k]} \notin R_{\mathrm{II}}(m_{[i]})$, because otherwise $m_{[k]} \in F_2$; if $m_{[k]} \in R_{\mathrm{III}}(m_{[i]})$, $(m_{[i]}, m_{[k]})$ causes the same hoist assignment to $m_{[i]}$ as $(m_{[i]}, m_{[j_1]})$ does; if $m_{[k]} \in R_{\mathrm{IV}} \cup R_{\mathrm{V}}(m_{[i]})$, then $(m_{[i]}, m_{[k]})$ does not cause any necessary hoist assignment to $m_{[i]}$. If $m_{[j_1]} \in R_{\mathrm{II}}(m_{[i]})$, then $m_{[k]} \in R_{\mathrm{II}} \cup R_{\mathrm{IV}} \cup R_{\mathrm{V}}(m_{[i]})$ according to P5(7), and therefore $(m_{[i]}, m_{[k]})$ does not cause different hoist assignment to $m_{[i]}$ from that caused by $(m_{[i]}, m_{[j_1]})$. If $m_{[j_1]} \in R_{\mathrm{V}} \cup R_{\mathrm{IV}} \backslash R_{\mathrm{II}}(m_{[i]})$, then $m_{[k]} \in R_{\mathrm{V}} \cup R_{\mathrm{IV}} \backslash R_{\mathrm{II}}(m_{[i]})$ according to P5(6), and therefore $(m_{[i]}, m_{[k]})$ does not cause any hoist assignment to $m_{[i]}$. In summary, $(m_{[i]}, m_{[k]})$ will never cause a different necessary hoist assignment to $m_{[i]}$ from that caused by $(m_{[i]}, m_{[j_1]})$.

Similarly, in case $m_{[k]} \in F_2$, $(m_{[i]}, m_{[k]})$ will never cause different hoist assignment to $m_{[i]}$ from that caused by $(m_{[i]}, m_{[j_2]})$.

In case $m_{[k]} \notin F_1 \cup F_2$, then $m_{[k]} \notin R_{\mathrm{I}} \cup R_{\mathrm{III}}(m_{[i]})$. Based on P3, $(m_{[i]}, m_{[k]})$ will not cause any hoist assignments to $m_{[i]}$.

Summarizing all of the above, we can conclude that all necessary hoist assignments to $m_{[i]}$ due to the move pairs $(m_{[i]}, m_{[k]})$ for all $k > i$, can be made by checking only two move pairs $(m_{[i]}, m_{[j_1]})$ and $(m_{[i]}, m_{[j_2]})$. □

Based on this proposition, all the necessary hoist assignments can be made by checking three move pairs for every move in the forward direction and then checking two move pairs for every move in the backward direction.

3.4. The Hoist Scheduling Procedure for a Type-1 Free T

Procedure 2 below is designed based on Proposition 6 to make all the necessary hoist assignments. It does this by checking P3(1, 2) for the individual moves and

Table 1. Process plan and processing times for the example.

Stage i	1	2	3	4	5	6	7	8	9	10	11	12	13
Tank s_i	2	5	8	10	13	12	11	9	7	4	6	3	1
Time τ_i	429	280	414	363	504	277	525	130	48	354	205	149	418

P3(3, 4, 5, 6) for move pairs in the forward direction, and then checking P3(3, 4, 7, 8) for move pairs in the backward direction. To ensure all necessary assignments for a complete cycle, the checking is done for two cycles. After the necessary assignments, the procedure performs feasibility checking and hoist scheduling.

Procedure 2. Hoist Assignment and Feasibility Checking for a Given Type-1 Free *T*

```
Step 1. Calculate $Y_i^s$ and $Y_i^e$, $i = 0, 1, \ldots, n$, for the given $T$; sequence the moves in the ascending order of $Y_i^s$ and get $Y_{[i]}^s$ and $Y_{[i]}^f$, $i = 0, 1, \ldots, 2n+1$. Let $F_1 = F_2 = \phi$, $i_1 = -1$, $i_2 = -1$.
Step 2. For $j = 0, 1, \ldots, 2n+1$:
    If $\min\{w_{s_{[j]}}, w_{s_{[j]+1}}\} < w_l + d$, then
        $F_1 = F_1 \cup m_{[j]}$, $i_1 = j$;
    If $\max\{w_{s_{[j]}}, w_{s_{[j]+1}}\} > w_r - d$, then
        $F_2 = F_2 \cup m_{[j]}$, $i_2 = j$;
    For $i = i_1, i_2, j-1$: If $i \neq -1$ and
        $m_{[j]} \in R_{\mathrm{I}}(m_{[i]})$, then $F_1 = F_1 \cup m_{[i]}$, $i_1 = i$,
        $F_2 = F_2 \cup m_{[j]}$, $i_2 = j$;
    For $i = i_1, i_2, j-1$: If $i \neq -1$ and
        $m_{[j]} \in R_{\mathrm{III}}(m_{[i]})$, then $F_2 = F_2 \cup m_{[i]}$, $i_2 = i$,
        $F_1 = F_1 \cup m_{[j]}$, $i_1 = j$;
    If $i_1 \neq -1$ and $m_{[j]} \in R_{\mathrm{IV}}(m_{[i_1]})$, then
        $F_1 = F_1 \cup m_{[j]}$, $i_1 = j$;
    If $i_2 \neq -1$ and $m_{[j]} \in R_{\mathrm{II}}(m_{[i_2]})$, then
        $F_2 = F_2 \cup m_{[j]}$, $i_2 = j$.
Step 3. If $F_1 \cup F_2 \neq \phi$, for $i = 2n, 2n-1, \ldots, 1, 0$:
    $j_1 = \min\{3n, j \mid j > i, m_{[j]} \in F_1\}$,
        $j_2 = \min\{3n, j \mid j > i, m_{[j]} \in F_2\}$;
    For $j = j_1, j_2$: If $j \neq 3n$ and $m_{[j]} \in R_{\mathrm{I}}(m_{[i]})$, then
        $F_1 = F_1 \cup m_{[i]}$, $F_2 = F_2 \cup m_{[j]}$;
    For $j = j_1, j_2$: If $j \neq 3n$ and $m_{[j]} \in R_{\mathrm{III}}(m_{[i]})$,
        then $F_2 = F_2 \cup m_{[i]}$, $F_1 = F_1 \cup m_{[j]}$;
    If $j_2 \neq 3n$ and $m_{[j_2]} \in R_{\mathrm{IV}}(m_{[i]})$, then
        $F_2 = F_2 \cup m_{[i]}$;
    If $j_1 \neq 3n$ and $m_{[j_1]} \in R_{\mathrm{II}}(m_{[i]})$, then
        $F_1 = F_1 \cup m_{[i]}$.
Step 4. If $F_1 \cap F_2 \neq \phi$, stop; $T$ is infeasible. Otherwise, assign all the moves in $F_1$ to $H_1$ and all the moves in $F_2$ to $H_2$; construct an optimal feasible schedule using the method described in the proof of Proposition 4. Stop.
```

Proposition 7. *Procedure 2 determines the feasibility of any given Type-1 free T and generates a hoist schedule in case it is feasible, with the computational complexity of $O(n \log n)$.*

Proof. Steps 2 and 3 in the procedure check conditions in Proposition 3 for two complete cycles forwards and backwards, to make all the necessary hoist assignments. Step 4 checks infeasibility in the assignments. If there is infeasibility, the procedure stops and the infeasibility is reported. If no infeasibility is identified for the assigned moves, a feasible schedule is constructed using the method described in the proof of Proposition 4. Therefore, for the given Type-1 free T, the procedure either finds the infeasibility or obtains a feasible schedule.

In the procedure, calculating the starting and ending times of the moves takes $O(n)$ time. Sequencing the moves takes $O(n \log n)$ time. The hoist assignment steps for each j in Step 2 and each i in Step 3 are constant and all the assignments in these two steps take $O(n)$ time. In Step 4, feasibility checking and linking the assigned moves to form two hoist paths take $O(n \log n)$ time. Therefore, the computational complexity for the whole procedure is $O(n \log n)$. □

3.5. An Example

We give an example here to demonstrate the application of Procedure 2. The example electroplating line has one loading/unloading station (station 0) and 13 processing tanks. Their positions are $w_i = i$, $i = 0, 1, \ldots, 13$. The process plan and processing times are listed in Table 1. Other parameters for the example problem are $w_l = 0$, $w_r = 13$, $d = 0.8$, $\nu = 0.2$, $\lambda = 0.4$, $\mu = 8$.

For a Type-1 free cycle length $T = 580$, the positions of the moves in the time-way diagram are shown in Figure 12. Applying Procedure 2, after the hoist assignments are made through forward checking in Step 2, we will get $F_1 = \{m_0, m_3, m_{10}, m_{13}, m_1\}$ and $F_2 = \{m_7, m_6, m_5, m_4\}$. The backward checking in Step 3 adds m_{11} and m_9 to F_1. After all these necessary hoist assignments, no move appears in both F_1 and F_2. Therefore, T will be verified to be feasible in Step 4. We can assign the free moves to H_2 and the complete assignment will be $F_1 = \{m_0, m_3, m_{10}, m_9, m_{11}, m_{13}, m_1\}$ and $F_2 = \{m_7, m_6, m_2, m_8, m_5, m_4, m_{12}\}$. Feasible paths for the hoists are then constructed and the result is shown in Figure 13.

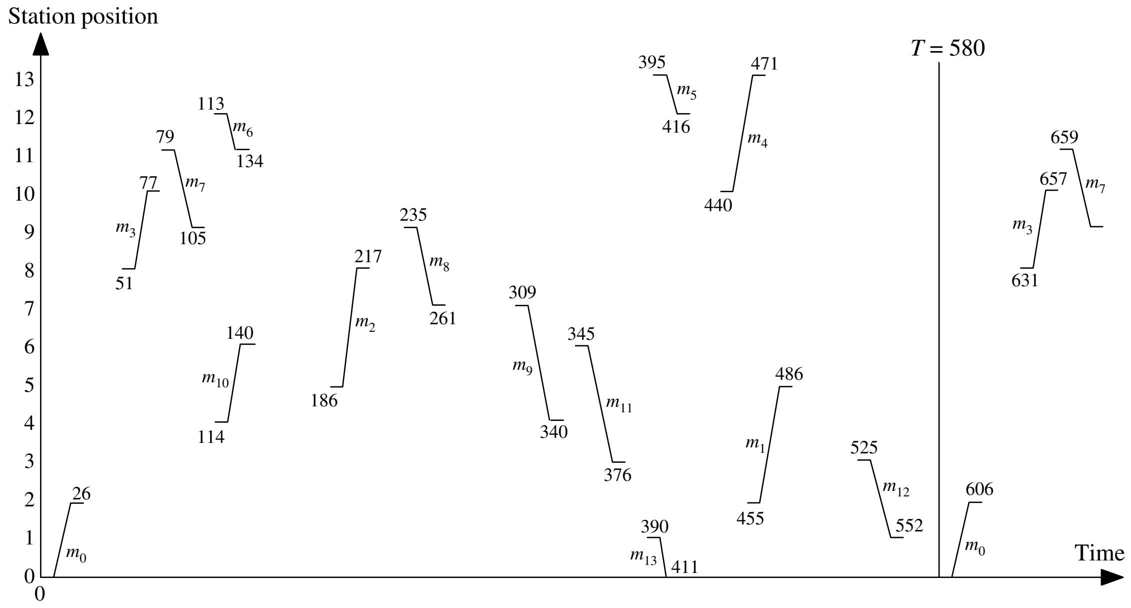
4. Further Thresholds and the Complete Algorithm

4.1. Second-Layer Thresholds

The feasibility properties of the points in a Type-1 free interval may not be the same. To check their feasibility efficiently, we need to find more thresholds at which the feasibility property may change.

From Propositions 3 and 4, we know that the feasibility property may change at a certain T value if the necessary conditions for the hoist assignments change at this T. We call these T values *second-layer thresholds*.

Figure 12. Positions of moves at a Type-1 free T for the example.



Definition 5. A cycle length, T, is called a second-layer threshold if, for any positive ϵ, there exists a positive number, $\delta < \epsilon$, such that at least one necessary hoist assignment is different for $T - \delta$ and $T + \delta$.

Based on the definition, we can identify the second-layer thresholds from the hoist assignment conditions for each pair of moves, m_i and m_j, $j > i$. These thresholds are T values that make the starting point of m_j on a boundary line between two regions of m_i or that make the starting point of m_i on a boundary between two regions of m_j.

When the starting point of m_j is on the boundary line between Regions I and II of m_i, from the expressions in the last section, we get

$$(w_{s_j} - w_{s_{i+1}})/\lambda = Y_j^s - Y_i^f = Z_j^s - Z_i^e - kT, \quad k = 1, 2, \ldots, j-i.$$

The corresponding cycle length is then

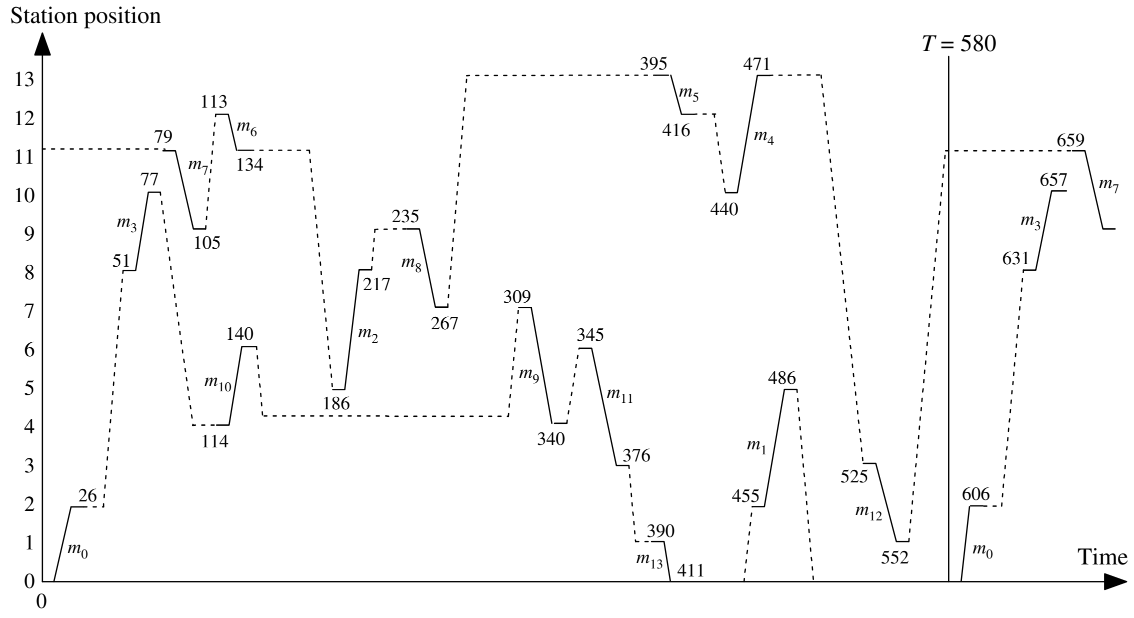
$$T = [Z_j^s - Z_i^e - (w_{s_j} - w_{s_{i+1}})/\lambda]/k, \quad k = 1, 2, \ldots, j-i.$$

Similarly, when the starting point of m_i is on the boundary line between Regions I and II of m_j, the corresponding cycle length is

$$T = [Z_j^e - Z_i^s + (w_{s_i} - w_{s_{j+1}})/\lambda]/k, \quad k = 1, 2, \ldots, j-i.$$

In the same way, we can obtain the second-layer thresholds for other boundary lines. A complete list of these

Figure 13. A feasible cyclic schedule for the example.



thresholds contains:

$$T_{ijk}^{1+} = [Z_j^s - Z_i^e - (w_{s_j} - w_{s_{i+1}})/\lambda]/k,$$
$$T_{ijk}^{1-} = [Z_j^e - Z_i^s + (w_{s_i} - w_{s_{j+1}})/\lambda]/k,$$
$$T_{ijk}^{2+} = [Z_j^s - Z_i^e - (w_{s_{i+1}} + d - w_{s_j})/\lambda]/k,$$
$$T_{ijk}^{2-} = [Z_j^e - Z_i^s + (w_{s_{j+1}} + d - w_{s_i})/\lambda]/k,$$
$$T_{ijk}^{3+} = [Z_j^s - Z_i^e - (w_{s_{i+1}} - w_{s_j})/\lambda]/k,$$
$$T_{ijk}^{3-} = [Z_j^e - Z_i^s + (w_{s_{j+1}} - w_{s_i})/\lambda]/k,$$
$$T_{ijk}^{4+} = [Z_j^s - Z_i^e - (w_{s_j} - w_{s_{i+1}} + d)/\lambda]/k,$$
$$T_{ijk}^{4-} = [Z_j^e - Z_i^s + (w_{s_i} - w_{s_{j+1}} + d)/\lambda]/k,$$

where $i = 0, 1, \ldots, n-1$, $j = i+1, \ldots, n$, $k = 1, 2, \ldots, j-i$.

In this list, T_{ijk}^{1+}, T_{ijk}^{2+}, T_{ijk}^{3+}, and T_{ijk}^{4+} correspond to the cases in which the starting point of m_j is on the boundary lines between Regions I and II, between Regions II and V, between Regions III and IV, and between Regions IV and V of m_i, respectively. T_{ijk}^{1-}, T_{ijk}^{2-}, T_{ijk}^{3-}, and T_{ijk}^{4-} correspond to the cases in which the starting point of m_i is on the boundary lines between Regions I and II, between Regions II and V, between Regions III and IV, and between Regions IV and V of m_j, respectively. It is easy to see that all the second-layer thresholds can be generated with computational complexity of $O(n^3)$.

Proposition 8. *The feasibility property of T in a Type-1 free interval may change only at the second-layer thresholds,* T_{ijk}^{1+}, T_{ijk}^{1-}, T_{ijk}^{2+}, T_{ijk}^{2-}, T_{ijk}^{3+}, T_{ijk}^{3-}, T_{ijk}^{4+}, T_{ijk}^{4-}, $i = 0, 1, \ldots, n-1$, $j = i+1, \ldots, n$, $k = 1, 2, \ldots, j-i$.

Proof. In determining the feasibility of a given Type-1 free T, Procedure 2 uses only the conditions in Propositions 3 and 4. The value (true or false) of such a condition changes only when the starting point of a move is on the boundary line of two regions of another move. The second-layer thresholds (the T values in the list) are all the values that make this happen. Therefore, the feasibility property of the problem may change only at second-layer thresholds. □

This proposition implies that, to solve the problem, we need to check the feasibility of only the first-layer thresholds and the second-layer thresholds in Type-1 free intervals. If we sequence these thresholds and check their feasibility one by one starting from the smallest one, then the first feasible threshold will be the optimal cycle length.

4.2. The Overall Procedure

With the developments in the previous sections, we can now integrate the results into a complete algorithm. Procedure 3 presents an overview description of the algorithm.

Procedure 3. The Overall Solution Procedure for the Problem

Step 1. Calculate T_0 and T^0; apply Procedure 1 to obtain the first-layer thresholds, α_p^l, α_p^u, $p = 1, \ldots, N_1$; calculate the second-layer thresholds.

Step 2. Sequence the first-layer and second-layer thresholds together in ascending order; delete the second-layer thresholds in the Type-1 infeasible intervals; and denote the remaining thresholds in ascending order as $\gamma_1, \gamma_2, \ldots, \gamma_N$; $T = \gamma_1$, $l = 1$.

Step 3. Apply Procedure 2 to T to make the necessary hoist assignments, check the feasibility, and if feasible, construct a hoist schedule.

Step 4. If T is feasible, stop. T is the optimal cycle length and the corresponding hoist schedule is an optimal schedule.

Step 5. If $l < N$, $l = l+1$, $T = \gamma_l$, go to Step 3. Otherwise, stop. There is no feasible solution to the problem.

Proposition 9. *The two-hoist, no-wait cyclic scheduling problem with fixed processing and transfer times is solvable in a computation time bounded by* $O(n^4 \log n)$.

Proof. The algorithm described in Procedure 3 skips Type-1 infeasible intervals for feasibility checking. In Type-1 free intervals, it checks every threshold. Because the feasibility property may change only at the thresholds, the algorithm does not miss any feasible solution in the searched intervals. The search starts from the lower bound of T, the value of T keeps increasing, and the algorithm stops when the first feasible solution is found. Therefore, the first feasible T found is the shortest cycle length, and the solution is optimal. If the algorithm stops after the entire range of $[T_0, T^0]$ is searched without obtaining a feasible solution, then the problem is infeasible. Consequently, the algorithm solves the problem in any case.

In this algorithm, each threshold is checked, at most, once for constructing a feasible schedule or identifying infeasibility (Procedure 2). From previous sections, we know that the computational complexity of Procedure 2 is $O(n \log n)$. The total number of thresholds including both layers is in the order of n^3. All the checking takes at most $O(n^4 \log n)$ time. All the thresholds are generated and sequenced with complexity of $O(n^3 \log n)$ before the checking starts. Therefore, the computational complexity of the entire algorithm is $O(n^4 \log n)$. □

5. Conclusions

In this paper, we have studied the two-hoist, no-wait cyclic scheduling problem in which the tank-processing times and the part transfer times are fixed parameters. The objective of the problem is to minimize the cycle length or, equivalently, to maximize the production rate. Based on the analysis of the problem properties, a polynomial algorithm was developed. The algorithm first identifies some threshold values of the cycle length so that feasibility many change only at these thresholds. It then searches for an optimal schedule by checking the feasibility of the threshold values. The computational complexity of the algorithm is $O(n^4 \log n)$.

One direction for further research may be to study the general multihoist, no-wait cyclic scheduling problem. The techniques developed in this paper will be useful. However, as the number of hoists increases, the conditions for hoist assignment and feasibility checking will be more and more complicated. For example, in the two-hoist case, there are only three possibilities in assigning a move to a hoist, i.e., necessary to Hoist 1, necessary to Hoist 2, and possible to either Hoist 1 or Hoist 2. When there are three hoists, the number of possibilities becomes much larger. Therefore, to solve the multihoist case efficiently, we will need additional new techniques, rather than extending the two-hoist results mechanically. Another direction for further research may be to develop heuristic solutions to the problem with time windows, using the no-wait algorithm as a subroutine.

Acknowledgment

This research was partially supported by a Direct Allocation Grant (DAG99/00.EG31) from Hong Kong Research Grant Council through Hong Kong University of Science and Technology.

References

Armstrong, R., L. Lei, S. Gu. 1994. A bounding scheme for deriving the minimal cycle time of a single-transporter N-Stage process with time-window constraints. *Eur. J. Oper. Res.* **78** 130–140.

Che, A., C. Chu, E. Levner. 2003. A polynomial algorithm for 2-degree cyclic robot scheduling. *Eur. J. Oper. Res.* **145** 31–44.

Chen, H., C. Chu, J. M. Proth. 1998. Cyclic scheduling of a hoist with time window constraints. *IEEE J. Robotic Automation* **14** 144–152.

Crama, Y., V. Kats, V. Van de Klundert, E. Levner. 2000. Cyclic scheduling in robotic flowshops. *Ann. Oper. Res.* **96** 97–124.

Kats, V., E. Levner. 1997. A strongly polynomial algorithm for no-wait cyclic robotic flowshop scheduling. *Oper. Res. Lett.* **21** 171–179.

Kats, V., E. Levner, L. Meyzin. 1999. Multiple-part cyclic hoist scheduling using a sieve method. *IEEE J. Robotic Automation* **15** 704–713.

Lamothe, J., M. Correge. 1995. A dynamic heuristic for the real time hoist scheduling problem. *Proc. 1995 INRIA/IEEE Sympo. on ETFA'95*, Vol. 2. IEEE Comput. Soc. Press, Paris, France, 161–168.

Lei, L. 1993. Determining the optimal starting times in a cyclic schedule with a given route. *Comput. Oper. Res.* **20** 807–816.

Lei, L., T. J. Wang. 1989. A proof: The cyclic hoist scheduling problem is NP-complete. Working paper 89/16, Rutgers University, Newark, NJ.

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

Lei, L., T. J. Wang. 1994. Determining optimal cyclic hoist schedules in a single-hoist electroplating line. *IIE Trans.* **26** 25–33.

Liu, J., Z. Zhou. 1998. A heuristic method for cyclic scheduling of two hoists without overlapping. *Proc. 3rd Annual Internat. Conf. Indust. Eng. Theories, Appl. Practice*, Hong Kong, pn250.1–pn250.6.

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

Ng, W. C. 1996. A branch and bound algorithm for hoist scheduling of a circuit board production line. *Internat. J. Flexible Manufacturing Systems* **8** 45–65.

Phillips, L. W., P. S. Unger. 1976. Mathematical programming solution of a hoist scheduling program. *AIIE Trans.* **8** 219–225.

Song, W., Z. B. Zabinsky, R. L. Storch. 1993. An algorithm for scheduling a chemical processing tank line. *Production Planning Control* **4** 323–332.

Sriskandarajah, C., N. G. Hall, H. Kamoun. 1998. Scheduling large robotic cells without buffers. *Ann. Oper. Res.* **76** 287–321.

Yang, G. W., D. P. Ju, W. M. Zheng, K. Lam. 2001. Solving multiple hoist scheduling problems by use of simulated annealing. *Ruan Jian Xue Bao J. Software* **12** 11–17.

Yin, N. C., Y. Yih. 1992. Crane scheduling in a flexible electroplating line: A tolerance-based approach. *J. Electronics Manufacturing* **2** 137–144.