OR Spectrum 29:445–470 (2007)
DOI 10.1007/s00291-006-0040-9

REGULAR ARTICLE

Ada Che · Chengbin Chu

Cyclic hoist scheduling in large real-life electroplating lines

Published online: 13 May 2006

Abstract This paper addresses cyclic scheduling of a single hoist in large real-life electroplating lines, where a part visits some processing tanks more than once and multiple duplicate tanks are used at some production stages having long processing times. We present a formal analysis of the problem and propose an efficient branch-and-bound algorithm. The developed analytical properties allow us to considerably eliminate dominated or infeasible solutions in the branch-and-bound procedure. Computational results on benchmark and real-life instances show that the algorithm is very efficient in scheduling large electroplating lines.

Keywords Hoist scheduling · Cyclic scheduling · Production systems · Electroplating lines

1 Introduction

1.1 Motivation

Many industrial processes involve multi-stage production lines, where material handling is performed by a computer-controlled hoist or robot. A typical application of such processes is an automated electroplating line for processing printed circuit boards (PCBs), as shown in Fig. 1. Such a production line is usually

A. Che (✉)
School of Management, Northwestern Polytechnical University,
Xi'an 710072, People's Republic of China
E-mail: ache@nwpu.edu.cn

C. Chu
ISTIT, Université de Technologie de Troyes, 12 Rue Marie Curie, BP 2060,
10010 Troyes Cedex, France

C. Chu
Hefei University of Technology, Tunxi Road 193, Hefei,
230009 Anhui, People's Republic of China
E-mail: chu@utt.fr

composed of a sequence of chemical processing stages. A chemical or plating treatment is performed on the part at each processing stage, such as acid activating, copper plating, rinsing, etc. During the process, each part must be successively soaked at each stage for a period of time, which must fall within the prescribed time window. Each tank can process only one part at a time. There is no buffer in the line. A computer-controlled hoist is used to move the part from one tank to the next. Effective scheduling of hoist movements is critical in achieving high throughput from these systems. This problem is commonly known as the hoist scheduling problem (Che et al. 2002; Che and Chu 2005; Chen et al. 1998; Kats and Levner 1998; Lei and Wang 1989; Lim 1997; Liu et al. 2002; Mak et al. 2002; Manier 1994, Manier and Bloch 2003; Ng 1996; Phillips and Unger 1976; Shapiro and Nuttle 1998; Sun et al. 1994; Varnier et al. 1997).

1.2 Literature review

Most previous research on hoist scheduling problems studied *basic* electroplating lines, where there is a one-to-one correspondence between the processing stages and the chemical tanks in the systems (Che et al. 2002; Chen et al. 1998; Lei and Wang 1994; Lim 1997; Phillips and Unger 1976; Shapiro and Nuttle 1998; Sun et al. 1994). However, real-life electroplating lines are often more complex than the basic system (Che and Chu 2005; Liu et al. 2002; Ng 1996; Varnier et al. 1997). The following are two common extensions:

1. Multi-function tank (Liu et al. 2002; Phillips and Unger 1976). More than one production stage may share one physical tank to save the facility cost. This is often the case for rinsing or drying tanks. Such a tank is called a *multi-function tank*. The part visits a multi-function tank more than once.

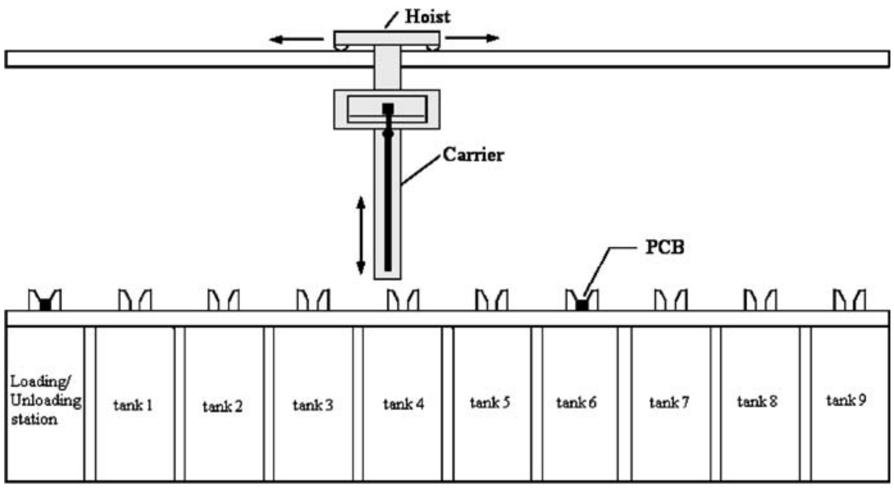

Fig. 1 An automated electroplating line for processing PCBs

2. Multi-tank stage (Che and Chu 2005; Liu et al. 2002). Production stages with long processing times can create severe bottlenecks in an electroplating line. To eliminate such bottlenecks and increase throughput, it is a common practice to add a group of duplicate tanks for these stages. A stage with multiple duplicate tanks is called a *multi-tank stage*.

Electroplating lines with multi-function tanks and multi-tank stages are called *extended* electroplating lines or *extended* systems (Liu et al. 2002). In real industrial environment, electroplating lines are usually operated cyclically for simplicity of implementation and ease of management. In such a cyclic production system, the hoist is programmed to perform a fixed sequence of moves repeatedly. Each repetition of the sequence of moves is called a *cycle*. The duration of a cycle is the *cycle time*. The cycle time measures the throughput of a production system. This paper addresses cyclic scheduling of a large real-life electroplating line with multi-function tanks and multi-tank stages. Particularly, we consider *simple* cyclic schedules with identical parts, where exactly one part enters and one part leaves the production line within a cycle. The criterion considered in this paper is cycle time minimization, which is consequently equivalent to maximizing throughput.

The simple cyclic single-hoist scheduling problem for the basic system was proved to be NP-complete by Lei and Wang (1989). This problem has been widely studied in the literature (Che et al. 2002; Chen et al. 1998; Lei and Wang 1994; Lim 1997; Phillips and Unger 1976; Shapiro and Nuttle 1998; Sun et al. 1994), while little work has been done for the extended system. Phillips and Unger (1976) presented the first integer programming model for the system with multi-function tanks. Other researchers (for example, Lei and Wang 1994; Ng 1996; Shapiro and Nuttle 1998) proposed branch-and-bound algorithms for the system with multi-tank stages. Recently, Liu et al. (2002) presented a mixed-integer programming model for the extended system and solved the MIP model using commercial optimization software CPLEX. Their algorithm can solve the problem with no more than 20 processing stages. Che and Chu (2005) developed a polynomial algorithm for a special system, where the parts' processing times are given constants, i.e., zero-width time windows.

Most studies in the literature assume that the time required for the loaded hoist to transport a part from one tank to the next is a given constant, where a *no-wait* constraint is imposed requiring that as soon as the operation is completed in a tank, the part must be immediately removed from the tank and transported to the next one without any delay. Ng (1996) noted that flexible move times longer than constant ones can lead to a shorter cycle time. This is, in fact, equivalent to allowing the hoist to pause during the transportation of a part from one tank to the next. Appendix A gives an example to illustrate the fact that flexible move times longer than constant ones can lead to a shorter cycle time. In this paper, flexible move times are considered, i.e., the hoist is allowed to pause during the transportation of a part from one tank to the next.

The main contribution of the paper is as follows. First, we perform a formal analysis of the problem and propose an efficient branch-and-bound algorithm. The developed analytical properties allow us to considerably eliminate dominated or infeasible solutions and greatly speed up the algorithm. As will be shown in later section, even for two large real-life electroplating lines with 31 processing tanks, our approach can find the optimal solution in reasonable computation time, while the

MIP approach proposed by Liu et al. (2002) can generally solve the problem with no more than 20 processing tanks. Second, we provide a more simplified formulation of the multi-function tank capacity constraint than that proposed by Liu et al., as will be described later. Third, this paper considers and solves several practical and benchmark problems in the literature. For one benchmark instance, we have found a better solution than that reported in the literature, and we report for the first time, to our knowledge, the optimal solution for two other large problems in the literature.

1.3 Related problems

This paper is devoted to the hoist scheduling problem. But the readers should note that a very similar problem of scheduling a material handling robot exists in cluster tools for semiconductor manufacturing (Kim et al. 2003; Lee et al. 2004; Lee and Lee 2006; Perkinson et al. 1996). A typical cluster tool consists of a number of wafer processing chambers and a wafer handling robot. As is the case in the hoist scheduling problem, the processing time of wafers in a chamber should be controlled within a given time window in most cases, especially for the low-pressure chemical vapor deposition process. Otherwise, the wafer will be subject to quality problems due to residual gases and heat (Kim et al. 2003). Modern cluster tools usually have two extended features. One is that parallel chambers to perform the same processing step are often used to balance the workloads among processing stages or steps (Kim et al. 2003; Lee et al. 2004; Perkinson et al. 1996). Another is that a wafer may visit a processing chamber more than once. Such a chamber is called a revisited or reentrant chamber (Kim et al. 2003; Lee and Lee 2006; Perkinson et al. 1996). We note that a revisited chamber and parallel chambers in cluster tool scheduling, respectively, correspond to a multi-function tank and duplicated tanks in the context of hoist scheduling problems.

Note that a similar scheduling problem also exists in more general manufacturing cells without work-in-process buffers. In such manufacturing cells, the parts are allowed to wait on the machine infinitely upon completion of their processing. This problem is often called robotic cells scheduling problem in the literature (Crama and van de Klundert 1997; Ioachim and Soumis 1995; Kamoun et al. 1999; Matsuo et al. 1991; Sethi et al. 1992; Sriskandarajah et al. 1998). Detailed description and classification of robotic cells scheduling problem can be found in (Crama et al. 2000; Dawande et al. 2005; Hall 1999). The computational complexity of these and related problems can be found in Hall et al. (1998). All the related problems deal with scheduling of a production system in which material handling among machines is executed by robots or hoists. For cyclic scheduling of more general flow shop or job shop without robots, please refer to (Hall et al. 2002; Lee and Posner 1997; Lee 2000; McCormick et al. 1989).

This paper is organized as follows. The "Problem description and formulation" section describes and presents an analytical model for the single-hoist scheduling problem considered in this paper. Properties of optimal solutions are analyzed in the "Property analysis of the optimal solution" section. Based on the presented model, an efficient branch-and-bound algorithm is proposed in the "Branch-and-bound algorithm" section. The "Computational results" section presents computational results on some well-known benchmark and real-life instances. The "Conclusion" section concludes the paper.

2 Problem description and formulation

2.1 Definitions, notations and assumptions

The production line considered is composed of N processing stages, S_1, S_2, ..., S_N. Let stages S_0 and S_{N+1} be the loading station and the unloading station, respectively. A single type of part is to be processed in this production line. The part flow can be described as follows. After a part is removed from S_0, it is processed successively through stages S_1, S_2, ..., S_N, and finally leaves the system from S_{N+1}.

To define the problem, the following problem parameters are given.

N	The number of processing stages in the production line, not including the loading stage (stage 0) and the unloading stage (stage N+1)
K	The total number of tanks available in the line
$\boldsymbol{G}_i$	The set of indices of tanks used to execute the process of stage i, i=0, 1, ..., N+1. By definition, $\lvert\boldsymbol{G}_i\rvert$ is the number of available processing tanks at stage i. If $\lvert\boldsymbol{G}_i\rvert=1$, stage i has a single tank; Otherwise, stage i has $\lvert\boldsymbol{G}_i\rvert$ tanks. We assume $\lvert\boldsymbol{G}_0\rvert=1$ and $\lvert\boldsymbol{G}_{N+1}\rvert=1$, i.e., there are no duplicates for the loading and the unloading stations
a_i	The lower bound on the processing time of a part at stage i, i=0, 1, ..., N
b_i	The upper bound on the processing time of a part at stage i, i=0, 1, ..., N
move i	The hoist move of transporting a part from S_i to S_{i+1}, i=0, 1, ..., N. Move i consists of three operations: unloading a part from S_i, transporting the part from S_i to S_{i+1}, and loading the part into S_{i+1}
θ_i	The constant time required for the hoist to perform move i, not including the pausing time, i=0, 1, ..., N
$\delta_{i,j}$	The time for an empty hoist to travel from S_i to S_j, $0\le i, j\le N+1$. $\delta_{i,j}$ satisfies the triangular inequality.

The decision variables of the considered problem include:

T	The cycle time
s_i	The starting time of move i relative to the start of a cycle, i.e., the amount of time that elapses after the start of the cycle before move i is performed, i=0, 1, ..., N
w_i	The pausing time of the hoist during execution of move i, i=0, 1, ..., N. As a result, execution of move i takes θ_i+w_i, for any $0\le i\le N$
m_i	The actual number of duplicate tanks to be used at stage i, $1\le m_i\le \lvert\boldsymbol{G}_i\rvert$, i=1, 2, ..., N.

To facilitate the development of the mathematical model, we define the following auxiliary variables:

t_i	The processing time of a part at stage i, i=0, 1, ..., N
c_i	0–1 variable, c_i=0 if $s_i>s_{i-1}$, i.e., unloading of a part from stage i happens after loading a part into the stage within a cycle; c_i=1 if $s_i<s_{i-1}$, i.e., unloading of a part from stage i happens before loading another part into the stage within a cycle, i=0, 1, ..., N.

The physical meaning of the above variables is illustrated in Fig. 2. This figure gives a cyclic schedule for a system with three stages. The processing of parts at stages 1, 2, and 3 are performed in tanks 1, 2, and 3, respectively. Tank 0 represents

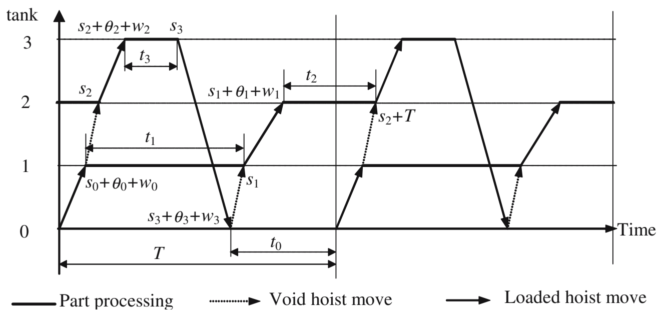

Fig. 2 A cyclic hoist schedule for an electroplating line with three tanks

the loading and unloading station. For this example, we have c_1=0, c_2=1, c_3=0, i.e., at the beginning of a cycle, tanks 1 and 3 are empty, while tank 2 contains a part introduced in the previous cycle.

As shown in Fig. 2, without loss of generality, we assume s_0=0, i.e., move 0 happens exactly 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 s_0>0, we can change the origin of the time axis such that s_0=0. With this assumption, c_0 is always equal to 1. As shown in Fig. 2, for this example, tank 0 is the loading/unloading station. For such line, for which the loading and the unloading stations are the same, because the loading and the unloading operations are often performed manually in real industrial environment, there must be sufficient time between unloading and loading operations on the same loading/unloading station. To formulate this constraint, we assume a fictitious processing time at stage 0, denoted by t_0, which is bounded from below by a_0 and from above by b_0. By definition, a_0 is the minimum time interval between unloading and loading operations on the same loading/ unloading station. In most cases, b_0 is set to +∞. The physical meaning of t_0 is illustrated in Fig. 2.

Before proceeding, let us introduce some basic ideas behind a multi-tank stage. For a multi-tank stage, each part is processed in only one of the duplicate tanks, and exactly one part is sent to and one part is removed from a multi-tank stage in a cycle. Hence, if there are m_i tanks used for stage i, then each of these duplicate tanks will process one part in m_i cycles.

Figure 3 shows an example of multi-tank stage i with m_i=3 (s_i>s_{i-1}). In the first cycle, a part is moved into the upper tank, while the other two tanks are processing

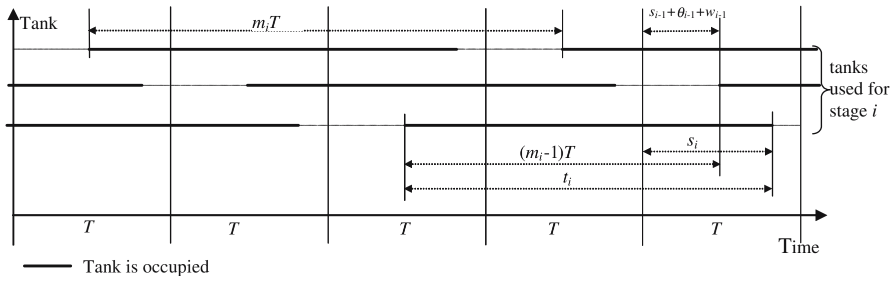

Fig. 3 Multi-tank stage i with m_i=3

parts that were moved into them in previous cycles. Later in this cycle, the middle tank completes its processing and a new part is moved into the middle tank in the second cycle. At a later time in the second cycle, the lower tank completes processing, and another part is moved into that tank in the third cycle. Later in the third cycle, the upper tank completes processing. The state in the fourth cycle is the same as that in the first cycle. For this example, each duplicate tank processes one part in three cycles.

When several duplicate tanks are used at stage i, the hoist move times $\delta_{i,j}$, θ_i, θ_{i-1} vary from cycle to cycle, depending on which tanks are used in the cycle. As we consider a cyclic schedule, to keep the schedule of all cycles the same, $\delta_{i,j}$, θ_i, θ_{i-1} must be chosen as the largest value they can assume for any duplicate tank for stage i, as is the case in all the previous research addressing multi-tank stages. Hence, $\delta_{i,i}$ becomes non-zero for any multi-tank stage i. It is the time for the hoist to travel between the two most distant duplicate tanks for stage i. In practical electroplating lines, the duplicate tanks for a stage are usually placed very closely, even a group of duplicate tanks correspond to a physical tank with the production capacity larger than one. Considering this fact, this approximation is quite reasonable.

In this paper, we assume that a multi-function tank cannot be a duplicate tank at the same time. This assumption is quite reasonable, as such a tank can be divided into two separate tanks to avoid the multi-function problem. For any two stages i and j using a multi-function tank, i.e., $\boldsymbol{G}_i=\boldsymbol{G}_j$, without loss of generality, we assume that $i \geq j+2$.

2.2 Mathematical model

In this section, we formulate our problem as a linear programming problem (LPP) provided that the sequence of hoist movements and the number of duplicate tanks used at each stage are given. In this case, the optimal values of the decision variables T, s_i, w_i, i=0, 1, ..., N, can be derived by solving this LPP. Then, in the "Branch-and-bound algorithm" section, a branch-and-bound procedure will be proposed to enumerate the sequence of hoist movements and the number of duplicate tanks used at each stage.

The hoist scheduling problem considered in this paper can be formulated as follows, as will be described in more detail later.

Problem P: Minimize T

subject to

Time window constraints on processing times

$$a_i \leq t_i \leq b_i, \text{ for all } i = 1, 2, \cdots, N. \tag{1}$$

$$t_i = (m_i - 1)T + s_i + c_i T - s_{i-1} - \theta_{i-1} - w_{i-1}, 1 \leq m_i \leq |G_i|, i = 1, 2, \cdots, N. \tag{2}$$

$$a_0 \leq T - s_N - \theta_N - w_N \leq b_0 \text{ if } G_{N+1} = G_0. \tag{3}$$

Hoist traveling time constraints

$$s_i + \theta_i + w_i + \delta_{i+1,j} \le s_j, \text{ for all } s_i \le s_j, i \ne j, i,j = 1,2,\cdots,N. \tag{4}$$

$$s_i \ge s_0 + \theta_0 + w_0 + \delta_{1,i}, \text{ for all } i = 1,2,\cdots,N. \tag{5}$$

$$s_i + \theta_i + w_i + \delta_{i+1,0} \le T, \text{ for all } i = 0, 1, \cdots, N. \tag{6}$$

Multi-function tank constraints

$$\begin{aligned} &s_i + \theta_i + w_i + \delta_{i+1,j-1} + \theta_{j-1} + w_{j-1} + t_j \le s_j,\\ &\text{for all } G_i = G_j, s_i \le s_j, i \ne j, i,j = 1,2,\cdots,N. \end{aligned} \tag{7}$$

$$\begin{aligned} &s_j + \theta_j + w_j + \delta_{j+1,i-1} + \theta_{i-1} + w_{i-1} + t_i \le T + s_i,\\ &\text{for all } G_i = G_j, s_i \le s_j, i \ne j, i,j = 1,2,\cdots,N. \end{aligned} \tag{8}$$

Non-negativity constraints

$$T \ge 0, s_i \ge 0, w_i \ge 0, \text{ for all } i = 0, 1, \cdots, N. \tag{9}$$

The objective of problem P is to minimize the cycle time. Constraint (1) means that the processing times of parts at stages must fall into the prescribed time windows. Constraint (2) gives the processing times of parts. When deriving the processing time of parts at stage i, depending on the values of c_i, two cases should be considered.

Case 1 c_i=0, i.e., $s_i > s_{i-1}$. In this case, unloading of a part from stage i happens after loading a part into the stage within a cycle. As mentioned above, if there are m_i tanks used for stage i, then each of the duplicate tanks will process one part in m_i cycles. Hence, in this case, a part entering stage i in any cycle l will leave from stage i in cycle (l+m_i−1). In Fig. 3, for the multi-tank stage with m_i=3, the part starting its processing on the upper tank in the first cycle finishes the processing and leaves from stage i in the third cycle. Note that a part enters stage i at time $s_{i-1}+\theta_{i-1}+w_{i-1}$, which is the ending time of move i−1, relative to the start of any cycle, and the part leaves from stage i at time s_i, as shown in Fig. 3. So, we have

$$t_i = (m_i - 1)T + s_i - s_{i-1} - \theta_{i-1} - w_{i-1}, 1 \le i \le N. \tag{10}$$

Case 2 c_i=1, i.e., $s_i < s_{i-1}$. In this case, unloading of a part from stage i happens before loading another part into the stage within a cycle. Similarly, as each duplicate tank processes one part in m_i cycles, a part entering stage i in any cycle l will leave from stage i in cycle (l+m_i). Hence, we have

$$t_i = m_i T + s_i - s_{i-1} - \theta_{i-1} - w_{i-1}, 1 \leq i \leq N. \quad (11)$$

In view of Eqs. (10) and (11), in either case, we have (2).

Constraint (3) ensures the time window constraint for stage 0, if the loading and the unloading stations are the same. As mentioned above, as the loading and the unloading operations are often performed manually in real industrial environment, there must be sufficient time between unloading and loading operations on the same unloading/loading station. Note that the transportation of a part from stage N to stage N+1 (i.e., move N) finishes at time $s_N+\theta_N+w_N$, and the loading of another part from stage 0 (i.e., move 0) after move N takes place at time T, as can be observed from Fig. 2. Therefore, we have $t_0=T-s_N-\theta_N-w_N$. As a consequence, we have Eq. (3).

Constraint (4) ensures that the empty hoist has sufficient time to travel between successive moves. For any pair of moves (i, j), if move j is performed after move i, i.e., $s_i \leq s_j$, then the empty hoist should arrive at S_j no later than s_j upon completion of move i. Note that move i finishes at time $s_i+\theta_i+w_i$, and the time for the empty hoist to travel from S_{i+1} to S_j is $\delta_{i+1,j}$. We thus have Eq. (4). Constraint (5) ensures the hoist traveling time constraints between move 0 and other moves, as we assume s_0=0. Constraint (6) says that the hoist has sufficient time to return to S_0 for the beginning of the next cycle upon completion of moves.

Relations (7) and (8) are concerned with the multi-function tank constraints. Due to the fact that each tank can process at most one part at a time, there must be sufficient time interval between the processing of any two stages i and j using the same multi-function tank. Due to the uniqueness of the hoist for material handling, if move j is performed after move i, as shown in Fig. 4, then move j can start only after the following operations are completed: (1) the hoist performs move i; (2) the hoist travels to S_{j-1} and transports a part from S_{j-1} to S_j (performs move j−1); and (3) processing of the part at S_j. Thus we have Eq. (7). Similarly, relations (8) are concerned with the multi-function tank constraints between move j and move i of the *next* cycle, as shown in Fig. 4. Note that this formulation of the multi-function

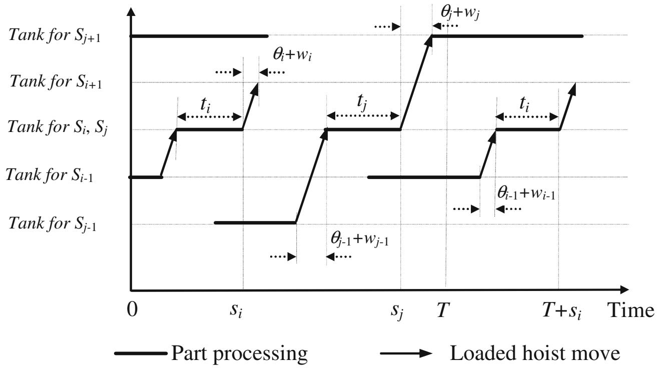

Fig. 4 Multi-function tank constraints

tank capacity constraint is more general and simplified than that proposed by Liu et al., as their formulation has to consider four possible conflicting cases for any two stages using the multi-function tank.

3 Property analysis of the optimal solution

In this section, we perform a property analysis for the optimal solution of the problem. We develop lower bounds on the optimal cycle time, lower and upper bounds on the optimal number of duplicate tanks to be used at a multi-tank stage, and upper bound on the number of parts processed simultaneously in a production line. The developed lower bounds and upper bounds will be used to eliminate dominated or infeasible solutions in the branch-and-bound procedure proposed in the next section.

3.1 Lower bounds on the optimal cycle time

As the hoist has to perform all the moves during a cycle, a lower bound of the cycle time is given by

$$LB_1 = \sum_{i=0}^{N} (\theta_i + \beta_i), \tag{12}$$

where β_i is the shortest time for the hoist to be ready for the next move upon completion of move i. When computing β_i, depending on the next move executed immediately after move i, two cases are possible.

Case 1 if the hoist performs move i+1 immediately after move i, then the hoist has to wait at stage i+1 for completion of processing of the part. In this case, we have β_i=$t_{i+1} \geq a_{i+1}$.

Case 2 if the hoist performs move j such that $j \neq i+1$ immediately after move i, then $\gamma_i \geq \min_{\substack{j \neq i+1 \\ 0 \leq j \leq N}} \delta_{i+1,j}$, which is the shortest time required for the empty robot to travel to perform any move after move i.

In short, we have $\beta_i = min\left\{\min_{\substack{j \neq i+1 \\ 0 \leq j \leq N}} \delta_{i+1,j}, \; a_{i+1}\right\}$. We note that a similar lower bound was given in (Lee et al. 2004; Perkinson et al. 1996) for scheduling of cluster tools, where the move/traveling times of the robot are assumed to be a constant and the second argument in the above formula of computing β_i was not considered, as in cluster tools scheduling, the processing times of wafers are usually longer than the move/traveling times of the robot. Hence, the lower bound presented in this paper is more general in this sense.

As mentioned above, if there are m_i tanks used for stage i, then each of the duplicate tanks will process one part in m_i cycles. As each tank can process at most one part at a time, T must be large enough to guarantee that the processing of two successive parts in the same tank will not overlap in time. We now derive a lower bound on m_iT. If $c_i = 0$, i.e., $s_i > s_{i-1}$, it follows from (10) that

$$m_iT = t_i + (T - s_i) + s_{i-1} + \theta_{i-1} + w_{i-1}, 1 \le i \le N.$$

According to Eqs. (5) and (6), this leads to

$$m_iT \ge t_i + (\theta_i + w_i + \delta_{i+1,0}) + (s_0 + \theta_0 + w_0 + \delta_{1,i-1}) + \theta_{i-1} + w_{i-1} \ge t_i + \theta_i + w_i + (\delta_{i+1,0} + \delta_{0,1} + \delta_{1,i-1}) + \theta_{i-1} + w_{i-1}.$$

As $\delta_{i,j}$ satisfies the triangular inequality, it is obvious that the above inequality can lead to:

$$m_iT \ge t_i + \theta_i + w_i + \delta_{i+1,i-1} + \theta_{i-1} + w_{i-1}, \text{for all } i = 1, 2, \cdots, N. \tag{13}$$

Similarly, If $c_i=1$, i.e., $s_i < s_{i-1}$, Eq. (11) can also lead to Eq. (13). In either case, Eq. (13) must hold. Due to $w_i \ge 0$ for any $0 \le i \le N$, it follows from Eq. (13) that

$$m_iT \ge t_i + \theta_i + \delta_{i+1,i-1} + \theta_{i-1}, \text{ for all } i = 1, 2, \cdots, N. \tag{14}$$

From Eqs. (1) and (14), another lower bound of the cycle time can be obtained by

$$LB_2 = \max_{1 \le i \le N} \frac{a_i + \theta_i + \delta_{i+1,i-1} + \theta_{i-1}}{|G_i|}. \tag{15}$$

We note that a similar lower bound was given in Lee et al. (2004) and Perkinson et al. (1996) for scheduling of cluster tools, where the move/traveling times of the robot are assumed to be a constant. The lower bound presented here is more general in this sense.

If stages τ_1, τ_2..., τ_l using the same multi-function tank, due to the multi-function tank capacity and the uniqueness of the hoist for material handling, another lower bound of the cycle time is

$$LB_3 = \sum_{i=1}^{l} \left(\theta_{\tau_i - 1} + a_{\tau_i} + \theta_{\tau_i + 1} + \min_{\substack{1 \le j \le l \\ j \ne i}} \delta_{\tau_i + 1, \tau_j} \right). \tag{16}$$

This relation comes from the fact that the processing of stages τ_1, τ_2..., τ_l in the multi-function tank cannot be overlapped in time and when the hoist performs moves $\tau_1 - 1$, $\tau_2 - 1$, ..., $\tau_l - 1$ and moves τ_1, τ_2..., τ_l, the multi-function tank must be empty due to the uniqueness of the hoist. Perkinson et al. (1996) gave a rough bound on the cycle time for cluster tools with reentrant flow. Once again, the lower bound presented here is more general.

3.2 Lower and upper bounds on the optimal number of duplicate tanks

We first derive a lower bound on the optimal number of duplicate tanks. It follows from Eqs. (1) and (14) that

$$m_i \geq \frac{a_i + \theta_i + \delta_{i+1,i-1} + \theta_{i-1}}{T} \geq \frac{a_i + \theta_i + \delta_{i+1,i-1} + \theta_{i-1}}{\overline{T}}, \text{for all } i = 1, 2, \cdots, N,$$

where $\overline{T}$ is the upper bound of T. Thus, a lower bound of m_i is

$$\underline{m}_i = \left\lceil \frac{a_i + \theta_i + \delta_{i+1,i-1} + \theta_{i-1}}{\overline{T}} \right\rceil, \text{for all } i = 1, 2, \cdots, N,$$

where the notation $\lceil x \rceil$ denotes the smallest integer greater than or equal to x.

We now derive an upper bound on the optimal number of duplicate tanks. As mentioned above, if there are m_i tanks used for stage i, then each tank at the stage will process one part in m_i cycles. Hence, for a given value of T, an upper bound of m_i must exist. To be more specific, If c_i=0, i.e., s_i>s_{i-1}, it follows from Eq. (10) that

$$(m_i - 1)T = t_i - s_i + s_{i-1} + \theta_{i-1} + w_{i-1}, 1 \leq i \leq N. \tag{17}$$

As s_i >$s_{i\text{-}1}$, it follows from Eq. (4) that

$$s_i \geq s_{i-1} + \theta_{i-1} + w_{i-1} + \delta_{i,i}. \tag{18}$$

From Eqs. (17) and (18), we must have

$$(m_i - 1)T \leq t_i - \delta_{i,i}, \text{for all } i = 1, 2, \cdots, N. \tag{19}$$

Similarly, if c_i=1, i.e., s_i<s_{i-1}, Eq. (11) can lead to Eq. (19). In either case, Eq. (19) must hold. From Eqs. (1) and (19), we have

$$m_i \leq \frac{(b_i - \delta_{i,i})}{T} + 1 \leq \frac{(b_i - \delta_{i,i})}{\underline{T}} + 1, \quad \text{for all } i = 1, 2, \ldots .N,$$

where $\underline{T}$ is the lower bound of T. Therefore, an upper bound of m_i is

$$\overline{m}_i = \left\lfloor \frac{(b_i - \delta_{i,i})}{\underline{T}} \right\rfloor + 1, \text{for all } i = 1, 2, \cdots, N,$$

where the notation $\lfloor x \rfloor$ denotes the integer part of x.

When deriving the lower and the upper bounds on the optimal number of duplicate tanks, we let $\underline{T}$ =max (LB$_1$, LB$_2$, LB$_3$). It is obvious that a feasible cycle can always be obtained by processing only one part in a cycle. Hence, an obvious upper bound of the cycle time is $\sum_{i=1}^{N} (a_i + \theta_{i-1,i})$. Initially, we let $\overline{T} = \sum_{i=1}^{N} (a_i + \theta_{i-1,i})$ and update $\overline{T}$ once we find a better feasible solution for the problem.

3.3 Upper bound on the number of parts processed simultaneously in the production line

Typically, more than one part is processed in a production line at any given time of a cycle. The number of parts that can be processed simultaneously in a production line depends on the input data for the production line. Let k be the number of parts processed simultaneously in the production line at the beginning of a cycle. If c_i=0, then loading a part into stage i happens before unloading of a part from the stage within the same cycle. This implies that one tank of stage i must be empty at the beginning of a cycle, as shown in Fig. 3. As a result, in this case (i.e., c_i=0), stage i contains (m_i−1) parts at the beginning of a cycle. Similarly, if c_i=1, then unloading a part from stage i happens before loading of another part into the stage within the same cycle. This implies that all duplicated tanks for stage i are occupied at the beginning of a cycle. Hence, in this case (i.e., c_i=1), stage i contains m_i parts at the beginning of a cycle time. In either case, stage i contains (c_i+m_i−1) parts at the beginning of a cycle. Thus, $k = \sum_{i=0}^{N} (c_i + m_i - 1) = \sum_{i=1}^{N} (c_i + m_i - 1) + 1$.

In the following, we will first derive the relations that any feasible k must satisfy, based on the mathematical model developed in the "Problem description and formulation" section. We then derive an upper bound on k from these relations and perform a complexity analysis. The developed upper bound on k will be used to eliminate infeasible solutions in the branch-and-bound algorithm in the next section.

Theorem 1 *The number of parts processed simultaneously in a production line at the beginning of a cycle satisfies the following inequalities:*

$$T \geq \max(LB_1, LB_2, LB_3). \tag{20}$$

$$kT \geq \sum_{i=0}^{N} \theta_i + \sum_{i=1}^{N} t_i + \delta_{N+1,0}. \tag{21}$$

$$kT \geq \sum_{i=0}^{N} \theta_i + \sum_{i=1}^{N} t_i + a_0, \text{if} G_{N+1} = G_0. \tag{22}$$

$$(k-2)T \leq \sum_{i=1}^{N-1} \theta_i + \sum_{i=1}^{N} t_i - \delta_{1,N} - LB_1. \tag{23}$$

$$(k-1)T \le \sum_{i=0}^{N}\theta_i + \sum_{i=1}^{N} t_i + b_0 - LB_1, \text{if}\, G_{N+1} = G_0. \tag{24}$$

$$a_i \le t_i \le b_i, \text{for all}\, i = 1, 2, \cdots, N. \tag{25}$$

$$\overline{m}_i T \ge t_i + \theta_i + \delta_{i+1,i-1} + \theta_{i-1}, \text{for all}\, i = 1, 2, \cdots N. \tag{26}$$

Proof From Eq. (2), we have

$$(c_i + m_i - 1)T = t_i - s_i + s_{i-1} + \theta_{i-1} + w_{i-1}, \text{for all } i = 1, 2, \cdots, N. \tag{27}$$

By summing both sides of Eq. (27) from i=1 to N, we obtain

$$(k-1)T = \sum_{i=1}^{N} t_i + \sum_{i=0}^{N-1}(\theta_i + w_i) - s_N. \tag{28}$$

Due to the fact that $s_N+\theta_N+w_N+\delta_{N+1,0} \le T$, it follows from Eq. (28) that

$$kT \ge \sum_{i=0}^{N}(\theta_i + w_i) + \sum_{i=1}^{N} t_i + \delta_{N+1,0}. \tag{29}$$

As $\sum_{i=0}^{N} w_i \ge 0$, Eq. (21) must hold according to Eq. (29).

According to Eq. (3), we must have

$$s_N + \theta_N + w_N + a_0 \le T, \text{if}\, G_{N+1} = G_0. \tag{30}$$

From Eqs. (28) and (30), we have Eq. (22).
Due to $s_N \ge \theta_0+w_0+\delta_{1,N}$, it follows from Eq. (28) that

$$(k-1)T \le \sum_{i=1}^{N-1}(\theta_i + w_i) + \sum_{i=1}^{N} t_i - \delta_{1,N}. \tag{31}$$

As LB_1 is the minimum move/traveling time of the hoist in a cycle, the following inequality must hold

$$\sum_{i=0}^{N} w_i \leq T - LB_1. \tag{32}$$

Relations (31) and (32) lead to (23).

From Eq. (3), we have

$$s_N \geq T - \theta_N - w_N - b_0. \tag{33}$$

Relations (28), (32), and (33) lead to Eq. (24). Relation (25) concerns the time window constraints. Relation (26) is derived from Eq. (14). We, thus, have Theorem 1.

Let $k_{\max}$ be the upper bound on the number of parts processed simultaneously in a production line at the beginning of a cycle. By Theorem 1, Eqs. (20, 21, 22, 23, 24, 25, 26) are constraints that any k must satisfy. In Eqs. (20, 21, 22, 23, 24, 25, 26), (T, t_1, t_2, ..., t_N) are decision variables of the problem and, thus, are unknown. For a given value of k, say k_0, if there exists at least one solution (T, t_1, t_2, ..., t_N) such that Eqs. (20, 21, 22, 23, 24, 25, 26) satisfy, then k_0 is said to be *feasible*. Otherwise, k_0 is said to be infeasible. As $k \in \{1, 2, ..., K\}$, $k_{\max}$ can be obtained by solving Eqs. (20, 21, 22, 23, 24, 25, 26) successively for $k = K$, $K-1$, ..., until the first feasible k is obtained. This value of k is $k_{\max}$. To determine $k_{\max}$, this needs to solve K linear inequality problems (20, 21, 22, 23, 24, 25, 26) in the worst case. For simplicity, the process of solving Eqs. (20, 21, 22, 23, 24, 25, 26) for a given value of k is called the feasibility detecting for this value of k. As described below, the feasibility detecting for a given value of k can be transformed into the cycle time evaluation problem in a bi-valued graph and, thus, can be implemented using a graph-based polynomial algorithm.

In what follows, we first give a definition for the cycle time evaluation problem in a bi-valued graph. A bi-valued graph can be defined by a two-tuple (V, E), where V and E are the set of vertices and the set of edges, respectively. Each edge $e \in E$ is associated with not only a length $l(e)$ but also a weight $w(e)$. Let $h(e)$ and $t(e)$ be the head and the tail of edge $e \in E$, respectively [i.e., edge e goes from vertex $t(e)$ to vertex $h(e)$]. Let π_v denote the potential of vertex $v \in V$. With this notation, in a bi-valued graph, edge e represents a constraint $\pi_{h(e)} - \pi_{t(e)} \geq l(e) - w(e)T$.

The cycle time of a bi-valued graph is defined as the optimal objective value T^* of the following problem, if it has a solution.

Problem CTE: Minimize T

subject to

$L(\gamma) - T \times H(\gamma) \leq 0$, $\forall \gamma \in \Gamma$, Γ being the set of directed circuits in a bi-valued graph, $ $<?A3B2 show [Equf]?><?A3B2 show [VP]?>T \geqslant 0.$$where $L(\gamma)$ and $H(\gamma)$, respectively, denote the length and the height of circuit γ. The length (resp. height) of a circuit is the sum of the lengths (resp. heights) of the arcs in the circuit. Note that if all circuits have positive heights, then $T^* = \max_{\gamma \in \Gamma} \{L(\gamma)/H(\gamma)\}$. However, for a bi-valued graph with negative heights, problem CTE may have no solutions, as there may exist circuits with negative heights.

Theorem 2 *The upper bound k_{max} can be derived in $O(K^2N^3)$ time in the worst case.*

Proof First, we show that Eqs. (20, 21, 22, 23, 24, 25, 26) can be represented by a bi-valued graph. For this purpose, by defining auxiliary variable $D_i \equiv \sum_{j=1}^{i} t_j$, $0 \le i \le N$, where D_0=0, Eqs. (20, 21, 22, 23, 24, 25, 26) can be equivalently expressed as

$$T \geq \max(LB_1, LB_2, LB_3). \tag{34}$$

$$D_0 - D_N \geq \sum_{i=0}^{N} \theta_i + \delta_{N+1,0} - kT. \tag{35}$$

$$D_0 - D_N \geq \sum_{i=0}^{N} \theta_i + a_0 - kT, \text{if} \, G_{N+1} = G_0. \tag{36}$$

$$D_N - D_0 \geq -\sum_{i=1}^{N-1} \theta_i + \delta_{1,N} + LB_1 + (k-2)T. \tag{37}$$

$$D_N - D_0 \geq -\sum_{i=0}^{N} \theta_i - b_0 + LB_1 + (k-1)T, \text{if} \, G_{N+1} = G_0. \tag{38}$$

$$D_i - D_{i-1} \geq a_i \, \text{for all} \, i = 1, 2, \cdots, N. \tag{39}$$

$$D_{i-1} - D_i \geq -b_i, \text{for all} \, i = 1, 2, \cdots, N. \tag{40}$$

$$D_{i-1} - D_i \geq \theta_i + \delta_{i+1,i-1} + \theta_{i-1} - \overline{m}_i T, \text{ for all } i = 1, 2, \cdots, N. \tag{41}$$

Each linear inequality among Eqs. (34, 35, 36, 37, 38, 39, 40, 41) can be expressed in the form of $D_j - D_i \geq l_{i,j} - h_{i,j}T$, where $h_{i,j}$ is an integer and $l_{i,j}$ is a real number. Due to this structure, we can construct a bi-valued graph for Eqs. (34, 35, 36, 37, 38, 39, 40, 41). The bi-valued graph corresponding to Eqs. (34, 35, 36, 37, 38, 39, 40, 41) contains N+1 vertices 0, 1, ..., N, the potentials of which are D_0, D_1, ..., D_N, respectively. Each linear inequality among Eqs. (34, 35, 36, 37, 38, 39, 40, 41), in the form of $D_j - D_i \geq l_{i,j} - h_{i,j}T$, corresponds to an edge from vertex i to vertex j with length $l_{i,j}$ and weight $h_{i,j}$. Thus, Eqs. (34, 35, 36, 37, 38, 39, 40, 41) are equivalently represented as

$$D_j - D_i \geq l_{i,j} - h_{i,j}T, \forall (i,j) \in E. \tag{42}$$

With the constructed directed graph, for a given value of k, any solution (T, t_1, t_2, ..., t_N) satisfies Eqs. (20, 21, 22, 23, 24, 25, 26) or equivalently Eqs. (34, 35, 36, 37, 38, 39, 40, 41), if and only if all the arcs on the graph satisfy Eq. (42). Let γ be a directed circuit on the graph, then by Eq. (42),

$$\sum_{\forall (i,j) \in \gamma} \left(D_j - D_i\right) \geq \sum_{\forall (i,j) \in \gamma} \left(l_{i,j} - h_{i,j}T\right).$$

As $\sum_{\forall (i,j) \in \gamma} \left(D_j - D_i\right) = 0$, we have $\sum_{\forall (i,j) \in \gamma} \left(l_{i,j} - h_{i,j}T\right) \leq 0$, for any γ.

Thus, for a given value of k, if there are no positive circuits in the associated directed graph for some T, then there must exist a feasible solution (T, t_1, t_2, ..., t_N) for Eqs. (20, 21, 22, 23, 24, 25, 26) or equivalently Eqs. (34, 35, 36, 37, 38, 39, 40, 41) and consequently this value of k is feasible, and vice versa. In fact, as soon as T is determined, the corresponding values of D_1, D_2, ..., D_N (or equivalently t_1, t_2, ..., t_N) can be determined by solving the longest path problem in the associated bi-valued graph (Chen et al. 1998). From this analysis, the feasibility of a given value of k is dependent on the value of T and independent of t_1, t_2, ..., t_N (or equivalently D_1, D_2, ..., D_N). The feasibility detecting for a given value of k using Eqs. (34, 35, 36, 37, 38, 39, 40, 41) can be, thus, described as: whether there exists a T, such that the corresponding graph contains no positive circuits. The latter problem is equivalent to the cycle time evaluation problem defined above. If problem CTE has a solution, then we must have found a feasible solution (T, t_1, t_2, ..., t_N) for a given value of k, and this value of k is feasible. Otherwise, if problem CTE has no solutions, any solution (T, t_1, t_2, ..., t_N) cannot lead to a graph without positive circuits, and, consequently, there exists no solutions for Eqs. (34, 35, 36, 37, 38, 39, 40, 41) for the given value of k. Such a value of k is infeasible.

From this analysis, the feasibility detecting for a given value of k using Eqs. (20, 21, 22, 23, 24, 25, 26) or equivalently Eqs. (34, 35, 36, 37, 38, 39, 40, 41) can be transformed to the cycle time evaluation problem in a bi-valued graph, which can be solved in $O(|V|^2|E|w_{\max})$ time in the worst case (Kats and Levner 1998), where $w_{\max}$ is the maximum absolute value of weights. For our problem, $|V|= N+1$, $O(|E|)=N$, $w_{\max}=K-1$. This means that Eqs. (20, 21, 22, 23, 24, 25, 26) can be solved in $O(KN^3)$ time in the worst case. To obtain $k_{\max}$, we need to solve at most K linear inequality problems (20, 21, 22, 23, 24, 25, 26). Consequently, $k_{\max}$ can be obtained in $O(K^2N^3)$ time in the worst case.

4 Branch-and-bound algorithm

In the "Mathematical model" subsection, we formulate our problem as a LPP, provided that the sequence of hoist movements and the number of duplicate tanks used at each stage are given. In this section, we propose a branch-and-bound procedure to enumerate the sequence of hoist movements and the number of duplicate tanks used at each stage. The feasibility of each enumeration and the optimal values of the decision variables T, s_i, w_i, i=0, 1, ..., N, can be derived by solving the corresponding relaxed LPP.

Each node in the search tree is associated with a set of partial precedence relations between hoist moves, denoted by O. For any pair of moves (i, j), if $s_i < s_j$, we say $(i, j) \in O$. Note that by the definition of c_i, $1 \le i \le N$, enumerating the values of c_i is equivalent to enumerating the sequence between moves $i-1$ and i.

As will be described in the following, the branching from a node at levels 0, 1, ..., $N-1$ is different from that from a node at levels N, $N+1$, ... The branching from a node at level $n-1$, for any $1 \le n \le N$, consists of enumerating the values of c_n and the values of m_n for processing stage S_n, as shown in Fig. 5. In Fig. 5, a node at level n, $1 \le n \le N$, with label (0, i) means that c_n=0 and m_n=i for stage S_n. Conversely, a node at level n, $1 \le n \le N$, with label (1, i) means that c_n=1 and m_n=i for stage S_n.

With this branching scheme, a node at level n, for any $1 \le n \le N$, is associated with a path from the root node to that node $\{(c_0, m_0), (c_1, m_1), ...(c_n, m_n)\}$ whose

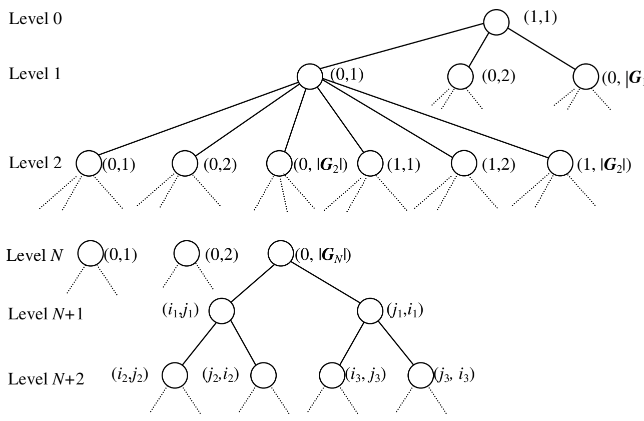

Fig. 5 Enumeration tree for the branch-and-bound algorithm

values are determined. Thus, a lower bound of the cycle time can be obtained by solving the following relaxed linear programming problem:

P_n: Minimize T
subject to

$$a_i \le s_i - s_{i-1} - \theta_{i-1} - w_{i-1} + (c_i + m_i - 1)T \le b_i, \text{ for all } i = 1, 2, \cdots, n.$$
$$s_i + \theta_i + w_i + \delta_{i+1,i-1} \le s_{i-1}, \text{ for all } c_i = 1, i = 1, 2, \cdots, n.$$
$$s_i \ge s_0 + \theta_0 + w_0 + \delta_{1,i}, \text{ for all } i = 1, 2, \cdots, n.$$
$$s_i + \theta_i + w_i + \delta_{i+1,0} \le T, \text{ for all } i = 0, 1, \cdots, n.$$
$$T \ge 0, s_i \ge 0, w_i \ge 0, \text{ for all } i = 0, 1, \cdots, n.$$

If there is no solution to this problem, or if there is a solution but the lower bound is greater than a known upper bound, then the corresponding node can be eliminated. Otherwise, the optimal value of the problem gives a lower bound of the cycle time.

As mentioned above, the branching from a node at level $n-1$, for any $1 \le n \le N$, consists of enumerating the values of c_n and the values of m_n for S_n. As c_n can take either 0 or 1 and $m_n \in \{1, 2, ..., |\boldsymbol{G}_n|\}$, a node at level $n-1$, $1 \le n \le N$, has at most $2|\boldsymbol{G}_n|$ sub-nodes. However, the lower bounds and upper bounds developed in the "Property analysis of the optimal solution" section can be used to eliminate some infeasible nodes. Let $k_n = \sum_{i=0}^{n} (c_i + m_i - 1)$. If a node is associated with a path $\{(c_0, m_0), (c_1, m_1), ...(c_n, m_n)\}$, such that k_n is greater than $k_{\max}$, then the node is eliminated. Similarly, if (c_n, m_n) related to a node at level n, $1 \le n \le N$, is such that $m_n < \underline{m}_n$ or $m_n > \overline{m}_n$, then the corresponding node can also be eliminated. As a multi-function tank can process one part at a time, for all stages using the tank, say stages $\tau_1, \tau_2..., \tau_l$, we must have $c_{\tau 1} + c_{\tau 2} + ... + c_{\tau l} = 1$. As a result, if $c_{\tau_k} = 1$ for some $1 \le k \le l$, we must have $c_j = 0$ for all $j = \tau_1, \tau_2..., \tau_l$ except τ_k.

The branching from a node at levels N, $N+1$, $N+2$, ..., is a binary search, which continues to enumerate the sequences of hoist moves. As mentioned above, each node in the tree is associated with a set of partial precedence relations between hoist moves, denoted by O. The lower bound of the cycle time corresponding to node O at levels N, $N+1$, $N+2$, ..., can be obtained by solving the following linear programming problem:

P_o: Minimize T
subject to

$$a_i \le s_i - s_{i-1} - \theta_{i-1} - w_{i-1} + (c_i + m_i - 1)T \le b_i, \text{ for all } i = 1, 2, \cdots, N.$$
$$s_i \ge s_0 + \theta_0 + w_0 + \delta_{1,i}, \text{ for all } i = 1, 2, \cdots, N.$$
$$s_i + \theta_i + w_i + \delta_{i+1,0} \le T, \text{ for all } i = 1, 2, \cdots, N.$$
$$s_i + \theta_i + w_i + \delta_{i+1,j} \le s_j, \text{ for all } (i,j) \in O.$$
$$s_i + \theta_i + w_i + \delta_{i+1,j-1} + \theta_{j-1} + w_{j-1} + t_j \le s_j, \text{ for all } (i,j) \in O, G_i = G_j.$$
$$s_j + \theta_j + w_j + \delta_{j+1,i-1} + \theta_{i-1} + w_{i-1} + t_i \le T + s_i, \text{ for all } (i,j) \in O, G_i = G_j.$$
$$T \ge 0, s_i \ge 0, w_i \ge 0, \text{ for all } i = 0, 1, \cdots, N.$$

The branching from a node at levels N, N+1, N+2, ..., is dependent on the solution of problem P_o, say [$T(O)$, $s(O)$, $w(O)$]. If [$T(O)$, $s(O)$, $w(O)$] satisfies constraints (4), (7), and (8), then a feasible schedule is obtained and no further branching from this node is needed, as any further branching from this node cannot lead to a better solution. Otherwise, there must be a pair of (i^*, j^*), such that one of the following conditions holds:

$$s_{i^*}(O) + \theta_{i^*} + w_{i^*}(O) + \delta_{i^*+1,j^*} > s_{j^*}(O), s_{i^*}(O) \le s_{j^*}(O)$$

$$s_{i^*}(O) + \theta_{i^*} + w_{i^*}(O) + \delta_{i^*+1,j^*-1} + \theta_{j^*-1} + w_{j^*-1}(O) + t_{j^*} > s_{j^*}(O), s_{i^*}(O) \le s_{j^*}(O), \text{ if } G_{i^*} = G_{j^*}$$

$$s_{j^*}(O) + \theta_{j^*} + w_{j^*}(O) + \delta_{j^*+1,i^*-1} + \theta_{i^*-1} + w_{i^*-1}(O) + t_{i^*} > T + s_{i^*}(O), s_{i^*}(O) \le s_{j^*}(O), \text{ if } G_{i^*} = G_{j^*}.$$

Such a pair of moves (i^*, j^*) is called a pair of overlapping ones. The branching from the node consists of enumerating two precedence relations between this pair of overlapping moves (i^*, j^*). Two subnodes and their associated partial precedence relation sets are generated by adding to O either a precedence constraint (i^*, j^*) or (j^*, i^*), as shown in Fig. 5, wherein a label, such as (i_1, j_1) for a node at levels N, N+ 1, N+2, ..., means that move i_1 happens before move j_1.

In the branch-and-bound algorithm, we use the depth first plus the best lower bound rule to select the node that should be considered next.

The computation of lower bounds in the branch-and-bound procedure requires the solution of specific linear programming problems (LPPs), i.e., P_n and P_o. We show that problems P_n and P_o can be also transformed into the cycle time evaluation problem in a bi-valued graph due to their specific structure. We take problem P_o as an example. By defining $Y_{i+1} \equiv s_i + w_i$, for all $0 \le i \le N$, problem P_o can be rewritten as

Problem P'_O : Minimize T

subject to

$$\begin{aligned}
& s_i - Y_i \ge a_i - (c_i + m_i - 1)T, \text{ for all } i = 1, 2, \cdots, N. \\
& Y_i - s_i \ge -b_i + (c_i + m_i - 1)T, \text{ for all } i = 1, 2, \cdots, N. \\
& s_0 - Y_{i+1} \ge \delta_{i+1,0} + T, \text{ for all } i = 1, 2, \cdots, N. \\
& s_j - Y_{i+1} \ge \delta_{i+1,j}, \text{ for all} (i, j) \in O. \\
& s_{j-1} - Y_{i+1} \ge \delta_{i+1,j-1} + c_j T, \text{ for all } (i, j) \in O, G_i = G_j. \\
& s_{i-1} - Y_{j+1} \ge \delta_{j+1,i-1} + (c_i - 1)T, \text{ for all } (i, j) \in O, G_i = G_j. \\
& Y_{i+1} - s_i \ge \theta_i, i = 0, 1, \cdots, N. \\
& s_i - Y_1 \ge \delta_{1,i}, i = 1, 2, \cdots, N.
\end{aligned}$$

Each linear inequality of Problem P'_O can be expressed in the form of $u_j - v_i \geq l_{i,j} - h_{i,j}T$, where $h_{i,j}$ is an integer and $l_{i,j}$ is a real number. Due to this structure, it can be easily seen that Problem P'_O can be transformed into the cycle time evaluation problem in a bi-valued graph and solved by using a graph-based polynomial algorithm.

5 Computational results

We have solved five benchmark instances available in the literature called Phillips, Black oxide 1, Black oxide 2, Copper, Zinc, and four real-life instances from actual industrial applications, Ligne 1, Ligne 2, Ligne 3, and Ligne 4. The data for these benchmark and real-life instances can be found in Manier 1994. Table 1 shows the instance sizes and features for the test instances. We will compare the proposed branch-and-bound algorithm with the mixed-integer programming (MIP) approach proposed by Liu et al. 2002 using commercial optimization software CPLEX (version 6.0.0). The computations were done on an HP J-5000.

Table 2 gives the optimal properties for the test instances. We find from Table 2 that the upper bound on the number of parts that can be processed simultaneously in a production line, $k_{\max}$, developed in this paper is very tight. For example, for Ligne 3 and Ligne 4, both with 31 tanks, if the number of parts processed in the production lines are greater than 17 and 11, respectively, then the corresponding solutions must be infeasible. This greatly reduces the size of the branch-and-bound tree.

The computational results on the test instances, using our approach, are listed in Table 3, wherein the columns "solution times", "B and B size", and "Average time for one LPP", respectively, give the computation time, the number of nodes of the branch-and-bound tree, and the average time required for solving one LPP for these instances. We find from Table 3 that even for large-size test instances like Ligne 3 and Ligne 4, with 18 and 35 processing stages, respectively, the solution times are

Table 1 Problem sizes and features for the test instances

Instances	Number of stages (N)	Number of tanks (K)	Number of multi-function tanks	Number of multi-tank stages
Phillips	12	12	0	0
Black oxide 1	11	12	0	1
Black oxide 2	11	12	0	1
Copper	11	17	0	2
Zinc	15	18	0	1
Ligne 1	12	12	0	0
Ligne 2	14	14	0	0
Ligne 3	18	31	1	4
Ligne 4	35	31	4	1

Table 2 Optimal properties for the test instances

Instances	K	k_{max}	LB_1	LB_2	LB_3	Optimum
Phillips	12	6	363	224	–	521
Black oxide 1	12	8	214.4	237.8	–	304.1
Black oxide 2	12	9	214.4	237.8	–	255.7
Copper	17	15	247.4	319.95	–	319.95
Zinc	18	11	274	429.48	–	435.85
Ligne 1	12	6	347	349	–	418
Ligne 2	14	6	401	712	–	712
Ligne 3	31	17	662	287.25	166	784.75
Ligne 4	31	11	857	880	491	1,585

within one hour and are considerably shorter than the days required for manual generation of a feasible schedule.

On the other hand, we would like to compare the optimal solutions obtained by using our approach with those reported in the literature to find out if new results have been obtained by this work. For the benchmark instances Phillips, Black oxide 1, Black oxide 2, Copper, and Zinc, the optimal cycle time found in this work is exactly the same as that reported in Shapiro and Nuttle 1998. For Ligne1 and Ligne 2, the optimal cycle times reported in Manier 1994 are 425 and 712, respectively. To the best of our knowledge, the optimal cycle time for Ligne 3 and Ligne 4 has never been reported in literature. Hence, we have found a better solution for Ligne 1 and report for the first time the optimal cycle time for Ligne 3 and Ligne 4. For interested readers, the optimal cyclic schedule for Ligne 1, Ligne 3, and Ligne 4 can be given by the authors upon request.

We also modeled these test instances using the mixed-integer programming (MIP) approach proposed by Liu et al. (2002) and solved them by using commercial optimization software CPLEX (version 6.0.0). Note that CPLEX also uses a branch-and-bound approach to solve the MIP problems. There is a sub-problem at each node of the tree, and each node is explored by solving the associated sub-problem. Table 4 gives the mixed-integer programming problem sizes and parameters for the test instances.

Table 3 Computational results on the test instances using our approach

Instances	Solution time (CPU s)	B and B size	Average time for one LPP (CPU ms)	Optimum in this work	Optimum in the literature
Phillips	0.59	3,692	0.16	521	521
Black oxide 1	0.27	1,696	0.16	304.1	304.1
Black oxide 2	0.78	4,095	0.19	255.7	255.7
Copper	0.08	579	0.15	319.95	319.95
Zinc	3.49	14,198	0.25	435.85	435.85
Ligne 1	0.96	5,463	0.76	418	425
Ligne 2	0.87	4,530	0.19	712	712
Ligne 3	34,066.00	28,678,163	1.18	784.75	Unknown
Ligne 4	3,308.80	3,271,935	1.00	1,585	Unknown

Table 4 MIP problem sizes and parameters for the test instances

Instances	*N*	*K*	Variables	0–1 variables	Constraints
Phillips	12	12	92	66	204
Black oxide 1	11	12	84	57	181
Black oxide 2	11	12	84	57	181
Copper	11	17	94	63	202
Zinc	15	18	146	109	313
Ligne 1	12	12	92	66	204
Ligne 2	14	14	121	91	266
Ligne 3	18	31	223	172	480
Ligne 4	35	31	675	595	1,408

The computational results on these instances, by using the CPLEX-based MIP approach, are listed in Table 5. For Ligne 3 and Ligne 4, computations were terminated because the system ran out of memory. In this case, the best feasible solutions are listed.

We can find from Tables 3 and 5 that both the computation times and the sizes of B and B tree using our approach are much less than those using the CPLEX-based MIP approach. For large-size test instances (e.g., Ligne3 and Ligne4), our approach found the optimal solutions in a relatively reasonable computation time, while CPLEX failed to find the optimal solutions because it ran out of memory. On the other hand, we are also concerned about the performance of the graph-based algorithm over classical linear program solver, such as the simplex method. It is observed from Tables 3 and 5 that the former are always much better than the latter, especially for solving the large-size linear programming problems (LPPs).

We note that the advantage of our approach over the MIP approach comes from two sides. One is the analytical properties developed in this paper, which allow us to considerably eliminate dominated or infeasible solutions in the branch-and-bound procedure. Another is that we use a graph-based polynomial algorithm to

Table 5 Computational results on the test instances using MIP approach

Instances	Solution time (CPU s)	B and B size	Average time for one LPP (CPU ms)	Optimum
Phillips	5.43	13,523	0.40	521
Black oxide 1	1.09	2,865	0.38	304.1
2	2.79	7,531	0.37	255.7
Copper	2.03	4,103	0.65	319.95
Zinc	79.49	121,143	0.66	435.85
Ligne 1	8.18	19,457	0.42	418
Ligne 2	4.76	8,156	0.58	712
Ligne 3	1,976.87[a]	2,194,634[a]	0.90[a]	945[a]
Ligne 4	25,956.22[a]	6,976,945[a]	3.72[a]	7,975[a]

[a] Terminated for running out of memory

solve the LPPs required for the computation of lower bounds in the branch-and-bound procedure due to their specific structure.

6 Conclusion

This paper proposed an efficient branch-and-bound algorithm for cyclic hoist scheduling in large real-life electroplating lines where a part visits some processing tanks more than once and multiple duplicate tanks are used at some production stages having long processing times. Computational results show that the algorithm is very efficient in scheduling large electroplating lines. We believe that this is mainly due to the developed analytical properties, which allow us to considerably eliminate dominated or infeasible solutions in the proposed branch-and-bound procedure, and the utilization of the graph-based polynomial algorithm.

Acknowledgements This work was in part supported by the Scientific and Technological Innovation Foundation from the Northwestern Polytechnical University and the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry, China.

Appendix A

An example in which flexible move times lead to a shorter cycle time

We use the following example, which is similar to the one in Ng 1996, to illustrate the fact that flexible move times longer than constant ones can lead to a shorter cycle time. There are four tanks in the line. The loading station and the unloading station are the same one (tank 0). The travel time for an empty hoist from tank *i* to tank *j* is $5|i-j|$ s, $0 \le i, j \le 3$. The constant times required for the hoist to transport a part from tank *i* to tank *i*+1 (including unloading a part from tank *i*, traveling from tank *i* to tank *i*+1, and loading the part onto tank *i*+1) for *i*=0, 1, 2, 3, are 10, 10, 10, and 20 s, respectively. The minimum processing times for tanks 0, 1, 2, and 3 are 40, 30, 30, and 20 s, respectively. The corresponding maximum processing times are 100, 60, 35, and 60 s.

With flexible move times, the optimal cycle time for this example is 105 s. The corresponding optimal hoist schedule is shown in Fig. 6. In this solution, the hoist pauses 5 s (from time instant 75 to 85) during its loaded travel from tank 1 to tank 2. As a result, the transportation of a part from tank 1 to tank 2 takes 15 s instead of

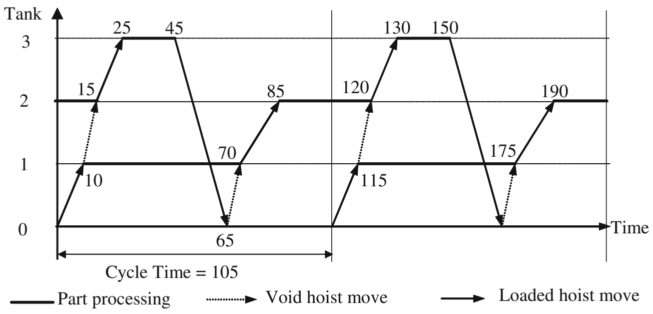

Fig. 6 The optimal cyclic hoist schedule with flexible loaded move times

10 s. For this example, if no pause was allowed during the hoist's loaded travel from tank 1 to tank 2, then the processing of the part in tank 2 will begin at time instant 80 instead of 85. Note that the part will be unloaded from tank 2 and complete the processing at time instant 120 in the *next* cycle. Therefore, the processing time for any part in tank 2 will be (120−80)=40. Thus, the processing time window in tank 2 will be violated. As a consequence, the corresponding schedule is infeasible. Thus, a feasible solution with flexible move times may be identified as an infeasible one with the constant travel time assumption. In fact, with the constant hoist move times, the optimal cycle time for this example is 110 s. Thus, flexible move times longer than constant ones lead to a shorter cycle time.

References

Che A, Chu C, Chu F (2002) Multicyclic hoist scheduling with constant processing times. IEEE Trans Robot Autom 18(1):69–80

Che A, Chu C (2005) A polynomial algorithm for no-wait cyclic hoist scheduling in an extended electroplating line. Oper Res Lett 33:274–284

Chen H, Chu C, Proth JM (1998) Cyclic scheduling of a hoist with time window constraints. IEEE Trans Robot Autom 14(1):144–152

Crama Y, van de Klundert J (1997) Cyclic scheduling of identical parts in a robotic cell. Oper Res 45(6):952–965

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

Dawande M, Geismar HN, Sethi SP, Sriskandarajah C (2005) Sequencing and scheduling in robotic cells: recent developments. J Sched 8(5):387–426

Hall NG (1999) Operations research techniques for robotic system planning, design, control and analysis. In: Nof SY (ed) Handbook of Industrial Robotics, vol. II, ch. 30. John Wiley, New York, 543–577

Hall NG, Kamoun H, Sriskandarajah C (1998) Scheduling in robotic cells: complexity and steady state analysis. Eur J Oper Res 109:43–65

Hall NG, Lee TE, Posner ME (2002) The complexity of cyclic shop scheduling problems. J Sched 5(4):307–327

Ioachim I, Soumis F (1995) Schedule efficiency in a robotic production cell. Int J Flex Manuf Syst 7:5–26

Kamoun H, Hall NG, Sriskandarajah C (1999) Scheduling in robotic cells: heuristic and cell design. Oper Res 47:821–835

Kats V, Levner E (1998) Cyclic scheduling of operations for a part type in an FMS handled by a single robot: a parametric critical-path approach. Int J Flex Manuf Syst 10:129–138

Kim JH, Lee TE, Lee HY, Park DB (2003) Scheduling analysis of time-constrained dual-armed cluster tools. IEEE Trans Semicond Manuf 16(3):521-534

Lee TE, Posner ME (1997) Performance measures and schedules in periodic job shops. Oper Res 45(1):72–91

Lee TE (2000) Stable earliest starting schedules for cyclic job shops: a linear system approach. Int J Flex Manuf Syst 12:59–80

Lee TE, Lee HY, Shin YH (2004) Workload balancing and scheduling of a single-armed cluster tool. In: Proceedings of the Fifth Asia Pacific Industrial Engineering and Management Systems (APIEMS) Conference, Gold Coast, Australia, pp 1–15

Lee HY, Lee TE (2006) Scheduling single-armed cluster tools with reentrant wafer flows. IEEE Trans Semicond Manuf (in press)

Lei L, Wang TJ (1989) A proof: the cyclic hoist scheduling problem is NP-hard. Working Paper #89-0016, Rutgers University

Lei L, Wang TJ (1994) Determining optimal cyclic hoist schedules in a single-hoist electroplating line. IIE Trans 26(2):25–33

Lim JM (1997) A genetic algorithm for a single hoist scheduling in the printed-circuit-board electroplating line. Comput Ind Eng 33:789–792

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

Mak RT, Gupta SM, Lam K (2002) Modeling of material handling hoist operations in a PCB manufacturing facility. J Electron Manuf 11(1):33–50

Manier MA (1994) Contribution à l'ordonnancement cyclique du système de manutention d'une ligne de galvanoplastie. Thèse de doctorat, Université de Franche-Comté, France

Manier MA, Bloch C (2003) A classification for hoist scheduling problems. Int J Flex Manuf Syst 15(1):37–55

Matsuo H, Shang JS, Sullivan RS (1991) A crane scheduling problem in a computer-integrated manufacturing environment. Manag Sci 17:587–606

McCormick ST, Pinedo ML, Shenker S, Wolf B (1989) Sequencing in an assembly line with blocking to minimize cycle time. Oper Res 37:925–935

Ng WC (1996) A branch and bound algorithm for hoist scheduling of a circuit board production line. Int J Flex Manuf Syst 8:45–65

Perkinson TL, Gyurcsik RS, McLarty PK (1996) Single-wafer cluster tool performance: an analysis of the effects of redundant chambers and revisitation sequences on throughput. IEEE Trans Semicond Manuf 9(3):384–400

Phillips LW, Unger PS (1976) Mathematical programming solution of a hoist scheduling program. AIIE Trans 8/2:219–225

Sethi SP, Sriskandarajah C, Sorger G, Blazewicz J, Kubiak W (1992) Sequencing of parts and robot moves in a robotic cell. Int J Flex Manuf Syst 4:331–358

Shapiro GW, Nuttle HW (1998) Hoist scheduling for a PCB electroplating facility. IIE Trans 20/2:157–167

Sriskandarajah C, Hall NG, Kamoun H (1998) Scheduling large robotic cells without buffers. Ann Oper Res 76:287–321

Sun T, Lai K, Lam K, So K (1994) A study of heuristics for bidirectional multi-hoist production scheduling systems. Int J Prod Econ 33:207–214

Varnier C, Bachelu A, Baptiste P (1997) Resolution of the cyclic multi-hoists scheduling problem with overlapping partitions. INFOR Inf Syst Oper Res 35(4):277–284