Cyclic Scheduling of a Hoist with time window constraints

Haoxun Chen, Chengbin Chu, Jean-Marie Proth

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Cyclic Scheduling of a Hoist with time window constraints

Haoxun CHEN - Chengbin CHU - Jean-Marie PROTH

N° 2307
Juillet 1994

PROGRAMME 5

ORDONNANCEMENT CYCLIQUE D'UN PONT ROULANT AVEC CONTRAINTES DE TYPE FENETRES TEMPORELLES

Haoxun CHEN*, Chengbin CHU**
and Jean-Marie PROTH***

Résumé : Cette publication propose un modèle et un algorithme de recherche de l'ordonnancement cyclique optimal d'un pont roulant avec contraintes de type fenêtres temporelles dans la fabrication de plaques portant des circuits imprimés. L'algorithme proposé est basé sur une approche de type séparation-évaluation qui demande que soient résolus des problèmes de programmation linéaire d'un type particulier. Ces problèmes sont équivalents à des problèmes d'évaluation des temps de cycle de graphes bi-valués. Des comparaisons avec les algorithmes de la littérature sont proposées.

Mots clés : Contraintes de type fenêtres temporelles, Ordonnancement, Graphe bi-valué, Temps de cycle, Séparation-évaluation.

* Professeur invité à INRIA-Lorraine du Systems Engineering Institute, Xi'an University, Xi'an, REPUBLIQUE POPULAIRE DE CHINE

** INRIA-Lorraine, Technopôle Metz 2000, 4 rue Marconi, 57070 Metz, FRANCE

*** INRIA-Lorraine, Technopôle Metz 2000, 4 rue Marconi, 57070 Metz, FRANCE, et Institute for Systems Research, University of Maryland, College Park, MD 20742, USA

CYCLIC SCHEDULING OF A HOIST WITH TIME WINDOW CONSTRAINTS

Haoxun CHEN*, Chengbin CHU**
and Jean-Marie PROTH***

Abstract: This paper proposes a model and a related algorithm for generating optimal cyclic schedules of hoist moves with time window constraints in a printed circuit board (PCB) electroplating facility. The algorithm is based on the branch and bound approach and requires the solution of a specific class of linear programming problems (LPPs). These LPPs are equivalent to the problems of the cycle time evaluation in bi-valued graphs. Computational experience is presented to compare the results obtained using this new algorithm with the ones proposed in the literature.

Keywords: Time window constraints, Hoist scheduling, Bi-valued graph, Cycle time, Branch-and-Bound.

* Invited Professor at INRIA-Lorraine from Systems Engineering Institute, Xi'an University, Xi'an, P.R. CHINA

** INRIA-Lorraine, Technopôle Metz 2000, 4 rue Marconi, 57070 Metz, FRANCE

*** INRIA-Lorraine, Technopôle Metz 2000, 4 rue Marconi, 57070 Metz, FRANCE, and Institute for Systems Research, University of Maryland, College Park, MD 20742, USA

1. INTRODUCTION

Many industrial processes employ a computer-controlled hoist for material handling. The hoist is programed to perform a fixed sequence of moves repeatedly. Each repetition of the sequence of moves is called a *cycle*. A typical application is an automated electroplating line for processing PCBs. This line consists of a sequence of tanks (see Figure 1). A chemical or plating treatment is performed on the PCBs in each tank. Treatments have to be performed in a given order. A tank contains at most one PCB at a time. Due to the nature of chemical treatments, the time a PCB spends in a tank is upper and lower bounded. This imposes *time window constraints* on the hoist moves. The hoist moves either from a tank to the next one to carry a PCB, or from one tank to any other if it is idle.

The fixed sequence of moves that the hoist performs in each cycle is defined by a *cyclic schedule*. In this paper, we consider *simple* cyclic schedules (Lei and Wang 1989a, Phillips and Unger 1976, Shapiro and Nuttle 1988). In a simple cyclic schedule, exactly one PCB is removed from each tank in a cycle; and therefore, one PCB enters and one PCB leaves the system in a cycle. The total time required by the hoist to complete a cycle is the *cycle time*. The objective of the hoist scheduling problem is to minimize the cycle time, which is equivalent to maximize the throughput rate.

This hoist scheduling problem has been proved to be NP-complete (Lei and Wang 1989b). All the existing approaches (Lei and Wang 1989a, 1991, Phillips and Unger 1976, Shapiro and Nuttle 1988, Hanen and Munier 1993) are branch-and-bound like. However, the efficiency of a branch-and-bound algorithm is very dependent on the quality of the lower bounds, the computation burden to obtain this lower bound, and the branching strategy .

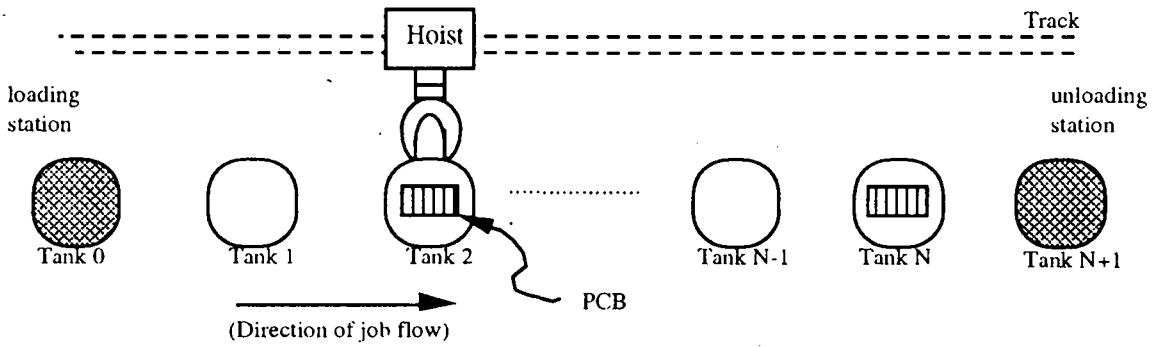


Fig. 1: Layout of an automated electroplating line for manufacturing PCBs

In this paper, we propose a model and a related algorithm for generating optimal cyclic schedules of hoist moves with time window constraints in a PCB electroplating facility. The algorithm is based on a branch and bound approach, which requires the solution of a specific class of linear programming problems.

Two branch-and-bound procedures are used. Procedure A implicitly enumerates all the possible initial PCB distributions in a cycle, knowing that : 1) the loading station always contains one PCB, and 2) the first move in a cycle consists of transporting a PCB from the loading station to tank 1. Some numerical constraints related to the processing times of the PCBs are used to reduce the number of initial PCB distributions. Procedure B implicitly enumerates the sequences of hoist moves for each potentially feasible initial PCB distribution.

The LPPs related to lower bound computation for the branch-and-bound algorithm in procedures A and B are transformed into cycle time evaluation problems in bi-valued graphs. These problems are solved using a graph-based iterative algorithm.

Computational experience on some benchmark problems is presented to compare the results obtained using this new algorithm with the results provided by the algorithms proposed in the literature.

The remainder of this paper is organized as follows. The hoist scheduling problem is formulated in Section 2. In Section 3, an upper bound of the maximal number of PCBs at the beginning of a cycle is proposed. The branch-and-bound algorithm is presented in Section 4. Section 5 is devoted to the solution of the linear programming problems. Section 6 gives computational results, and Section 7 is the conclusion.

2. PROBLEM FORMULATION

Consider a production process with N+2 stations (tanks), $S = \{S_0, S_1, S_2, ..., S_N, S_{N+1}\}$, where S_0, S_{N+1} are respectively the loading station and the unloading station, and a hoist that moves PCBs between stations (see Figure 2). For simplicity, the *move* of a PCB from S_i to S_{i+1} is called the move i, i=0,1,....,N. The times for the hoist to perform move i, denoted by θ_i^1, $0 \le i \le N$, and the time needed to travel from S_j to S_k, denoted by θ_{jk}^0, $0 \le j, k \le N+1$, are given. These times are supposed to be constant. Station S_i, $1 \le i \le N$, can process at most one PCB at a time, and each PCB must be processed in S_i in a time t_i which is lower bounded by a_i and upper bounded by b_i, i.e., $a_i \le t_i \le b_i$. The values of a_i and b_i, i=1,2,....,N are given and constant . They impose *time window constraints* on the moves. That is, given that a PCB is put in S_k at time t, the hoist must be ready to remove that PCB from S_k within $[t+a_k, t+b_k]$. In addition, we

assume that S_0 (the loading station) contains an infinite number of identical PCBs, that $a_0=0$ and $b_0=\infty$, that no PCB is allowed to wait between stations, and that no treatment can be preempted. As mentioned above, we only consider simple cyclic schedules.

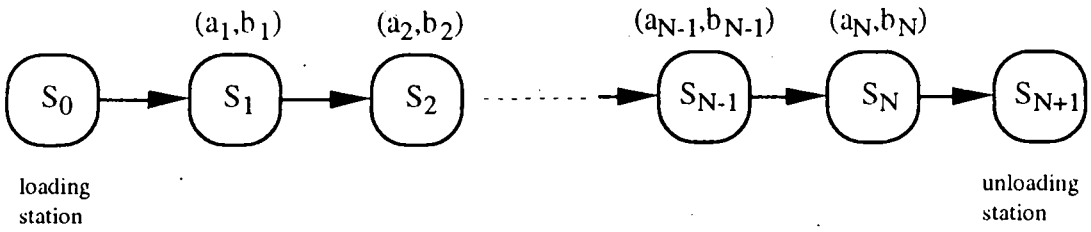


Fig. 2: The PCB flow through a system with N+2 stations.

As mentioned in the introduction, we only consider simple cyclic schedules. Simple cyclic schedules possess properties which n-cyclic schedules (n>1) generally do not. For a simple cyclic schedule with a cycle time x, the time between successive introductions of PCBs into the system is always x. Furthermore, one PCB is introduced into the system and another one leaves the system, and one PCB (not necessarily the same) is introduced in and removed from each tank during each cycle. Moreover, the treatment time in a given tank is the same for each PCB. Figure 3 illustrates a simple cyclic schedule for a 6-station system. The crosshatched areas correspond to moves, when the hoist is busy. The intervals between moves represent treatment times.

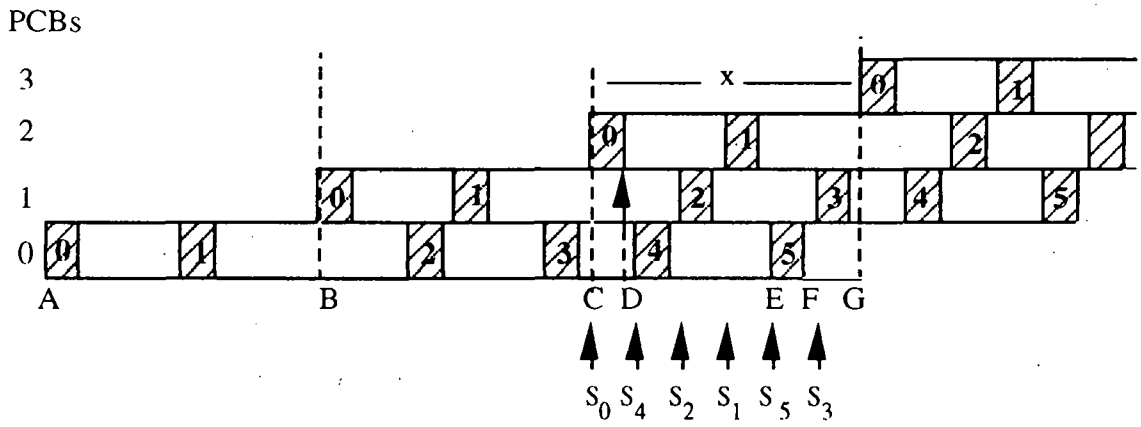


Fig. 3: A simple cyclic schedule for a 6-station system.

Let us consider a simple cycle the first move of which is the move 0 as shown in Figure 3 and denote by s_i the starting time of move i during the cycle, i=0,1,2,...,N, and by x the cycle time. We assume that $s_0=0$ (Note that if $s_0>0$, we can change the origin of the time axis such that $s_0=0$). According to the previous notations, $(x, \{s_i, i=0,1,...,N\})$ ((x,s) for short) defines a cyclic schedule. A cyclic schedule $(x, \{s_i, i=0,1,...,N\})$ is

feasible if and only if it satisfies the following constraints:

1. The time a PCB spends in tank S_i is lower bounded by a_i and upper bounded by b_i, i=1,2,...,N.

2. Stations S_i, i=1,2,...,N, must be empty when a PCB arrives at S_i.

3. There must be sufficient time for a hoist to travel between successive assigned tanks.

These three kinds of constraints are referred to as time window constraints, tank capacity constraints and hoist capacity constraint respectively.

Let $C = \{c_0,c_1,.....,c_N\}$, where $c_i = 1$ if the station i contains a PCB at the beginning of a cycle and 0 otherwise, and $K(C) = \sum_{i=0}^{N} c_i$. C is called the initial PCB distribution related to the cycle, and K(C) is the number of PCBs which are serviced by the hoist in the cycle. Note that c_0 is always equal to 1 since we assume that a PCB is always available at the loading station. In the example of Figure 3, we have K=3, $c_0=c_2=c_4=1$, $c_1=c_3=c_5=0$. Using the previous notations, the time window constraints can be formulated as:

$$a_i \le s_i + c_i x - s_{i-1} - \theta^1_{i-1} \le b_i, \qquad i=1,2,...,N \tag{2.1}$$

The tank capacity constraints can be formulated as

$$s_i + \theta^1_i + \theta^0_{i+1,i-1} \le s_{i-1}, \qquad \text{for any i with } c_i=1 \tag{2.2}$$

Let $\sigma=<l_0 l_1 .....l_{N-1} l_N>$ be the sequence of moves corresponding to the cyclic schedule. Then the hoist capacity constraint can be formulated by means of (2.3) and (2.4)

$$s_{l_i} + \theta^1_{l_i} + \theta^0_{l_i+1,l_{i+1}} \le s_{l_{i+1}}, \qquad i=0,1,.....,N-1 \tag{2.3}$$

$$s_i + \theta^1_i + \theta^0_{i+1,0} \le x, \qquad i=0,1,.....,N \tag{2.4}$$

Constraints (2.3) can be equivalently replaced by the following constraints

$$s_i + \theta^1_i + \theta^0_{i+1,j} \le s_j, \quad \text{if } s_i \le s_j, \quad i\neq j,\ i,\ j=0,1,.....,N \tag{2.3'}$$

Constraints (2.3') are obvious if S_j is the station visited after move i; otherwise, the

constraints still hold since the hoist will arrive at S_j later than the time it would arrive at S_j if S_j were the station visited after move i.

Constraints (2.4) complement constraints (2.3) for the last hoist move in the cycle.

Then, the hoist cyclic scheduling problem can be formulated as

P_{hcs}: min x

s.t. (2.1) to (2.4)

3. Upper Bound of the Maximal Number of PCBs

In this section, we give an upper bound of the maximal number of PCBs which can be serviced by the hoist in a cycle. The upper bound will be used to reduce the solution space of the branch and bound algorithm introduced in the next section.

Let k be the number of PCBs which are serviced by the hoist in a cycle and x the cycle time, the following inequalities hold:

$$x \geq \sum_{i=0}^{N} \theta_i^1 + \min_{i\in\{1,\ldots,N\}} \theta_{i,0}^0 \tag{3.1}$$

$$kx \geq \sum_{i=0}^{N} \theta_i^1 + \sum_{i=1}^{N} t_i + \theta_{N+1,0}^0 \tag{3.2}$$

$$(k-1)x \leq \sum_{i=1}^{N-1} \theta_i^1 + \sum_{i=1}^{N} t_i - \theta_{1,N}^0 \tag{3.3}$$

$$a_i \leq t_i \leq b_i, \qquad i=1,2,\ldots,N \tag{3.4}$$

$$t_i \leq x - \theta_{i-1}^1 - \theta_i^1 - \theta_{i+1,i-1}^0, \quad i=1,2,\ldots,N \tag{3.5}$$

where the variable t_i represents the treatment time of each PCB in station S_i, i=1,...,N.

Inequality (3.1) means that the cycle time is greater than or equal to the sum of the times necessary for the hoist to transport the PCBs and the minimal time for the hoist to come back to the loading station in a cycle.

Inequality (3.2) reflects the fact that the first introduced PCB will leave the system before the end of the k-th cycle. In Figure 3, this means that AG=kx=AF+FG,

$AF=\sum_{i=0}^{N} \theta_i^1 + \sum_{i=1}^{N} t_i$, $FG \geq \theta_{N+1,0}^0$ (Note that in Figure 3, N=5, k=3).

Inequality (3.3) reflects the fact that the first introduced PCB will leave the system during the k-th cycle and that at least the move N of the hoist will happen during the k-th cycle. In Figure 3, this means AC=(k-1)x=AF-CD-DE-EF, CD=θ_0^1, $\theta_{1,N}^0 \le$DE, EF= θ_N^1.

Inequality (3.5) means that the cycle time is greater than or equal to the sum of the times of move i-1 and move i, and the time for the hoist to travel from S_i to S_{i-1} (this result is easy to be found when one considers a cycle starting from move i).

Let K_{max} be the upper bound. Since $K \in \{1,2,...,N\}$, K_{max} can be obtained by solving inequalities (3.1) to (3.5) successively for k =N, N-1, ..., until the first solution is obtained. This solution is K_{max}. This needs to solve at most N-1 linear inequality problems.

4. BRANCH AND BOUND ALGORITHM

In this section, we introduce a branch and bound algorithm for generating an optimal cyclic schedule for hoist scheduling problems. The algorithm consists of two branch-and-bound procedures A and B . Procedure A implicitly enumerates all the possible initial PCB distributions in a cycle. Procedure B implicitly enumerates the sequences of hoist moves for each initial PCB distribution.

Let $C = \{c_0, c_1, ..., c_N\}$ be an initial PCB distribution. For each $k \le N$, $C_k = \{c_0, ..., c_k\}$ is called a partial initial (PCB) distribution. Let $K(Ck) = \sum_{i=0}^{k} c_i$.

In Procedure A, each node corresponds to a partial initial distribution. For a given partial initial distribution $C_k = \{c_0, ..., c_k\}$, a lower bound of the cycle time can be obtained by relaxing part of the time window constraints (2.1) and the tank capacity constraints (2.2) corresponding to the i's for which c_i is unknown, and by relaxing the hoist capacity constraint (2.3'). In other words, the lower bound can be obtained by solving the following linear programming problem:

P_{C_k}: min x

$$a_i \le s_i + c_i x - s_{i-1} - \theta_{i-1}^1 \le b_i, \qquad i \le k \tag{4.1}$$

$$s_i + \theta_i^1 + \theta_{i+1,i-1}^0 \le s_{i-1}, \qquad \text{for any } i \le k \text{ with } c_i = 1 \tag{4.2}$$

$$s_i + \theta_i^1 + \theta_{i+1,0}^0 \le x, \quad i=0,1,.....,N \qquad (4.3)$$

where constraints (4.3) are constraints (2.4).

If there is no solution to this problem, then the partial initial distribution cannot lead to a feasible cyclic schedule, and thus the corresponding node can be eliminated. Otherwise, the optimal objective value of this problem provides a lower bound of the cycle time.

Note that if Procedure A reaches a partial initial distribution C_k such that $K(C_k)$ is larger than K_{max}, then the node can be eliminated.

When Procedure A reaches a complete initial and potentially feasible PCB distribution C ($C=C_N$), if the solution $(x,\{s_0,s_1,...,s_N\})$ of the problem P_C satisfies constraints (2.3'), then the solution is an optimal cyclic schedule with respect to the given initial PCB distribution. Otherwise, the solution violates part of the constraint (2.3').

Procedure B looks for an optimal cyclic schedule for a given initial PCB distribution C ($C=C_N$). In Procedure B, each node is associated with a set of partial precedence constraints between two hoist moves, denoted by O, together with the constraints of problem P_C (this node is denoted by (C,O)). At the root node, $O = \varnothing$. The lower bound of the cycle time corresponding to this node can be obtained by solving the following linear programming problem:

$P_{C,O}$: min x

s.t. $s_i + \theta_i^1 + \theta_{i+1,j}^0 \le s_j$, if $(i,j)\in O$;

and constraints (4.1), (4.2) and (4.3) of P_C.

If $P_{C,O}$ has no solution, then the given set of partial precedence constraints cannot lead to a feasible solution, and thus the corresponding node can be eliminated. Otherwise, a lower bound of the cycle time is obtained.

If the solution of problem $P_{C,O}$ satisfies constraints (2.3'), a feasible cyclic schedule with respect to the given initial distribution C is obtained. Otherwise, two subnodes are derived from node (C,O). The two subnodes and their associated partial precedence constraint sets are generated according to the solution of problem $P_{C,O}$. That is, if $P_{C,O}$ has a solution, say (x(C,O),s(C,O)), and if (x(C,O),s(C,O)) satisfies constraints (2.3'), then we find a feasible cyclic schedule for the given initial distribution C. Otherwise,

there is a pair of (i^*,j^*) such that:

$$s_{i^*}(C,O)+\theta^1_{i^*}+\theta^0_{i^*+1,j^*} > s_{j^*}(C,O) \text{ and } s_{i^*}(C,O) \le s_{j^*}(C,O)$$

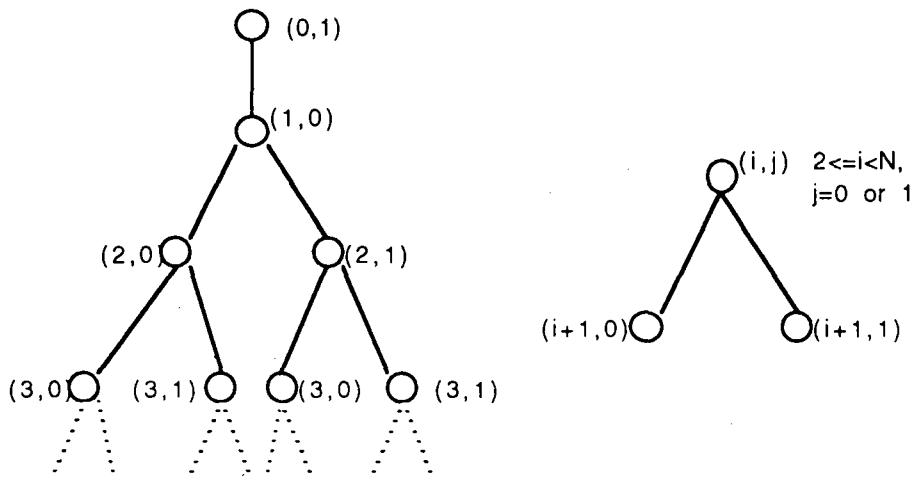
(in this case, moves i^*, j^* are called a pair of overlapping moves). Then, two partial precedence constraint sets associated with the two subnodes are generated by adding to O either a precedence constraint (i^*,j^*) or (j^*,i^*).

It should be noted that the number of elements in O is upper bounded by (N-1)N/2 - (N-1), where (N-1)N/2 is the maximal number of precedence constraints which can be assigned to N moves (move 1 to move N), and N-1 is the number of precedence constraints which are fixed by constraints of problem P_C.

The total number of nodes enumerated in Procedure B closely depends on the maximal number of elements in the partial precedence constraint sets reached by Procedure B. In practice, this maximal number is much smaller than (N-1)N/2 - (N-1) since the sequences of hoist moves are largely restricted by the time window constraints when the initial PCB distribution is specified.

In the following, we describe in detail procedures A and B in the branch and bound algorithm.

Enumeration Tree:



(a) enum. tree A (b) subnodes

Fig.4: The enumeration tree for procedure A of the branch and bound algorithm

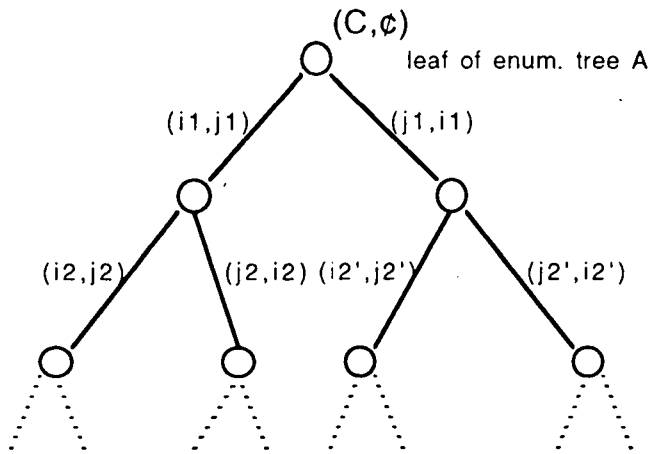


Fig. 5: The enumeration tree for procedure B of the branch and bound algorithm

Figure 4 (a) and Figure 5 show respectively the enumeration tree for procedure A and procedure B in the branch and bound algorithm (we will call them enumeration tree A and enumeration tree B respectively). Figure 4 (b) gives the general rule for creating subnodes from a node in the enumeration tree A. In enumeration tree A, a node with label (i,0) means that the station S_i is empty at the beginning of a cycle. Conversely, a node with label (i,1) means that the station S_i has a PCB at the beginning of the cycle. Note that since we assume that the first move of the cycle is move 0, station 1 should be empty at the beginning of the cycle. As a consequence, node (1,1) does not exist. The tree stops growing if the initial PCB distribution of all stations is assigned. If a partial initial distribution C_k related to a node is such that $K(C_k) = K_{max}$, then C_k can be completed to obtain a complete initial distribution by setting $c_i=0$ for all i>k, and the node does not generate any subnode furthermore.

The root node of enumeration tree B is a leaf of enumeration tree A (a node in an enumeration tree is called a leaf if the node has no subnode). Like enumeration tree A, the branching (the growth of subnodes) of a node in enumeration tree B is also a dichotomy. In the case when there exists two moves i and j which do not satisfy constraints (2.3'), we generate two new nodes by adding respectively precedence constraints (i,j) and (j,i) to the set of partial precedence constraints O. In Figure 5, a label beside a line linking a node to its subnode represents such a precedence constraint . The tree stops growing if the solution of the relaxed problem for a node does not exhibit overlapping moves or if the relaxed problem has no solution.

Lower Bounds:

Enumeration tree A

For any node (k, j_k) (j_k=0 or 1) in the enumeration tree A such that (0,1) ->(1,0) -> $(2,j_2)$ -> ... -> (k, j_k) is the path from the root of the tree to the node, then the lower bound corresponding to the node is obtained by solving the linear programming problem P_{C_k}, where $C_k = \{1,0,j_2,...,j_k\}$.

Enumeration tree B

If a node of enumeration tree B is derived from the root node $(C,\varnothing)$ (C is an initial PCB distribution) by successively adding precedence constraints (i_1,j_1), (i_2,j_2), ..., (i_l,j_l), then the lower bound corresponding to the node is obtained by solving the linear programming problem $P_{C,O}$, where $O = \{(i_1,j_1), (i_2,j_2), \ldots, (i_l,j_l)\}$.

Pruning Criteria:

In both procedures, the pruning conditions are:

1. Infeasiblity: the relaxed problem for a node has no solution;

2. Value dominance: Let x_R^α be the optimal objective value of the relaxed problem for a node α and $\bar{x}$ the objective value of a feasible solution to P_{hcs}. Then, if $x_R^\alpha \geq \bar{x}$, the node can be eliminated.

Node Selection:

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

5. Solution of the Relaxed Problems: A Graph-Based Algorithm

In the previous section, we proposed a branch and bound algorithm for generating an optimal cyclic schedule for hoist scheduling problems, where the lower bounds in procedures A and B are the solutions of specific linear programming problems (LPPs). In these LPPs (i.e., P_{C_k}, $P_{C,O}$ in Section 4), each linear inequality constraint can be equivalently formulated as $s_j - s_i + h_{ji}x \geq l_{ji}$ for $h_{ji} = 0$, 1 or -1, $l_{ji} \in \mathbb{R}$, where $\mathbb{R}$ is the set of real numbers.

In this paper, we propose a graph-based algorithm to solve this class of LPPs. This algorithm is based on the computation of the cycle time of bi-valued graphs including negative heights. First, we give the definition of bi-valued graphs and show that LPPs can be transformed into the cycle time evaluation of bi-valued graphs, and then we provide the graph-based algorithm. The convergence and computation complexity analysis of the graph-based algorithm can be found in Appendix.

5.1. Bi-Valued Graphs

Definition 5.1: A bi-valued graph is 4-tuple G =(V, A, l, h), where V ={1, 2, ..., n} is the set of vertices. A = $\{a^1, a^2, ..., a^m\}$ is the set of the arcs. l : A $\rightarrow \mathbb{R}$ defines the lengths of the arcs. h : A $\rightarrow \mathbb{R}$ defines the heights of arcs.

We assume that the origin and the extremity of any arc in G are distinct, i.e., there is no re-entrant arc in G, but G may have two arcs with the same origin and the same extremity.

In the following, we denote by o(a) (resp. e(a)) the origin (resp. extremity) of arc a. Arc a is called a forward (resp. backward) arc if o(a)<e(a), i.e., the label number of its origin is smaller than the one of its extremity (resp. o(a)>e(a), i.e., the label number of its origin is larger than the one of its extremity). B denotes the number of backward arcs, and $B' = \min \{B, n-1\}$. For each vertex k, $I_k = \{ a \mid e(a) = k \}$, $FI_k = \{ a \mid e(a) = k, o(a)<k \}$, $BI_k = \{ a \mid e(a) = k, o(a)>k \}$ denote the set of incident arcs, the set of forward incident arcs, the set of backward incident arcs respectively.

Let us denote the length (resp. height) of a path (circuit) π in the bi-valued graph G by $L(\pi)$ (resp. $H(\pi)$). The length (resp. height) is the sum of the lengths (resp. heights) of the arcs in this path (circuit).

Definition 5.2: The cycle time of the bi-valued graph G is defined as the optimal objective value x^* of the following problem if this problem has a solution

Problem P. Min x

subject to

$L(\gamma) - x \times H(\gamma) \leq 0$, for any $\gamma \in \Gamma$, where Γ is the set of elementary circuits (5.1)

$x \geq 0$. (5.2)

Note that if all the arcs of the bi-valued graph have non negative heights and all elementary circuits have positive heights, then $x^* = \max_{\gamma \in \Gamma}\{L(\gamma)/H(\gamma)\}$, where G is the set of elementary circuits.

It should be noticed that for a bi-valued graph with negative heights, the problem P may not have a solution, since there may exist circuits with negative heights.

5.2. Bi-Valued Ggraphs for Solving P_{C_k} and $P_{C,O}$

According to (Chrétienne 1984), any linear programming problem with an objective function x and whose linear constraints can be written as $s_j - s_i + h_{ji}x \geq l_{ji}$ ($h_{ji} \in \mathbb{Z}$, $l_{ji} \in \mathbb{R}$, and $\mathbb{Z}$ is the set of integers) can be transformed into the cycle time evaluation problem of a bi-valued graph. Therefore, the LPPs met in the computation of the lower bounds in the branch and bound algorithm (i.e., P_{C_k}, $P_{C,O}$ in Section 4) can be solved using the bi-valued graph defined as follows:

- $n = N+1$, $V = \{0,1,....N\}$;
- for each constraint $a_{i+1} \leq s_{i+1} - s_i - \theta_i^1 \leq b_{i+1}$, there is an arc from vertex i to vertex i+1 the length and the height of which are $a_{i+1}+\theta_i^1$ and 0 respectively, and an arc from vertex i+1 to vertex i the length and the height of which are $-b_{i+1}-\theta_i^1$ and 0 respectively.
- for each constraint $a_{i+1} \leq s_{i+1} + x - s_i - \theta_i^1 \leq b_{i+1}$, there is an arc from vertex i to vertex i+1 the length and the height of which are $a_{i+1}+\theta_i^1$ and 1 respectively, and an arc from vertex i+1 to vertex i the length and the height of which are $-b_{i+1}-\theta_i^1$ and -1 respectively.
- for each constraint $s_i + \theta_i^1 + \theta_{i+1,j}^0 \leq s_j$, there is an arc from vertex j to vertex i the length and the height of which are $\theta_i^1 + \theta_{i+1,j}^0$ and 0 respectively.
- for each constraint $s_i + \theta_i^1 + \theta_{i+1,0}^0 \leq x$ (i.e., $s_i - s_0 + \theta_i^1 + \theta_{i+1,0}^0 \leq x$), there is an arc from vertex i to vertex 0 the length and the height of which are $\theta_i^1 + \theta_{i+1,0}^0$ and 1 respectively.

5.3. An Algorithm for Evaluating the Cycle Time of Bi-Valued Graphs

In what follows, we develop an algorithm to check if the cycle time of a bi-valued graph exists, and possibly, to compute its value. This algorithm is based on the Ford-Bellman's algorithm used to check the existence of positive length circuits in an ordinary graph.

Assume that a lower bound of x^*, say LB, is known (Such a lower bound always exists, since 0 is a lower bound according to (5.2)). We can construct an ordinary graph G'_{LB} derived from G =(V, A, l, h) in which the length of any arc a is $l'_{LB}(a) = l(a) - h(a)\cdot LB$, i.e., $G'_{LB} = (V, A, l')$. If $L'_{LB}(\gamma)$ denotes the length of a circuit $\gamma \in \Gamma$ in G'_{LB}, then we have :

$$L'_{LB}(\gamma) = L(\gamma) - H(\gamma)\cdot LB \qquad (5.3)$$

If for any $\gamma \in \Gamma$, $L'_{LB}(\gamma) \le 0$, then $x^* = LB$. In other words, LB is the optimal cycle time in this case.

If there is a circuit γ^* such that $L'_{LB}(\gamma^*) > 0$, or equivalently $L(\gamma^*)-H(\gamma^*)\cdot LB > 0$, then two cases may happen. If $H(\gamma^*) \le 0$, then for any $x \ge LB$, we have $L(\gamma^*)-H(\gamma^*)\cdot x \ge L(\gamma^*)-H(\gamma^*)\cdot LB > 0$. This means that the problem P has no solution. If $H(\gamma^*) > 0$, according to (5.1), we have $x^* \ge L(\gamma^*)/H(\gamma^*) > LB$, then a better lower bound has been found, which is $L(\gamma^*)/H(\gamma^*)$. In this way, we can find the optimal cycle time or prove that the problem P has no solution.

Therefore, in order to get the optimal cycle time, it is necessary to detect if there is a circuit of positive length in an ordinary graph. This problem can be solved by a Ford-like algorithm as follows.

Ford-Like Algorithm FLA:

Step 1. $i := 0$; $\lambda(1) := 0$; $\lambda(k) := -\infty$, $\forall\ 1 < k \le n$.

Step 2. If $i > B'$, there is no positive length circuit, STOP.

Step 3. For k = 1, 2, ..., n

3.1. For any $a \in I_k$

If $\lambda(o(a)) + l'(a) > \lambda(k)$

begin

$la(k) := a$;

pc := CHECK_CIRCUIT(k);
if pc = TRUE, a positive-length circuit is found, STOP.
else $\lambda(k) := \lambda(o(a)) + l'(a)$.

end

Step 4. i := i + 1, GOTO Step 2.

Function CHECK-CIRCUIT(k):

Step 1. pc := FALSE, v := k.

Step 2. v := o(la(v)).

Step 3. If v = k then pc := TRUE, GOTO Step 5.

Step 4. If $v \neq 1$, GOTO Step 2.

Step 5. RETURN pc.

The proof of the correctness of algorithm FLA is in Appendix.

Using FLA, an algorithm to compute the cycle time of a bi-valued graph is as follows:

Cycle Time Evaluation Algorithm CTE:

Step 1. $LB = LB^0$, LB^0 is a lower bound of the minimum cycle time x^* of G.

Step 2. Check if there is a positive-length circuit in the ordinary graph G'_{LB} by using the algorithm FLA.

If not, then $x^* = LB$, stop; otherwise, denote this circuit by γ^*, and go to Step 3.

Step 3. If $H(\gamma^*) \leq 0$, then there is no feasible cycle time for G, stop; otherwise,

set $LB = L(\gamma^*)/H(\gamma^*)$, and go to Step 2.

The computation complexity of algorithm CTE in the worst case is $O(n^4 \cdot m^2)$. In the branch and bound algorithm, the lower bound of the cycle time related to a subnode is greater than or equal to the one related to its ascendent node. Thus, when we use algorithm CTE to solve the LPP related to the subnode, we can take the initial lower bound LB^0 as the lower bound related to its ascendent node. This makes algorithm CTE very efficient in practice.

It should be noted that the algorithm CTE can also be used to solve any LPP which can be transformed into the cycle time evaluation of a bi-valued graph.

6. COMPUTATIONAL RESULTS

We have applied our algorithm to five problems. These problems were solved on an IBM 6091/19 computer. Our algorithm uses the depth first search plus backtracking rule to select the next node and the algorithm CTE to solve all relaxed problems. The results are summarized in Table 1.

Table 1. Summary of Test Results

Problem tested	Number of tanks	Upper bound of K	Number of LPP's solved	Solution time (CPU seconds)	Average time for one LPP (CPU millisec)	Optimal cycle time
Philu	13	6	991	0.52	0.525	521
Ligne 1*	13	6	1341	0.76	0.567	393
Ligne 2*	14	5	622	0.25	0.402	712
Black Oxide 1*	12	7	570	0.24	0.421	281.9
Black Oxide 2*	12	7	243	0.08	0.329	279.3

The problems Philu, Black Oxide 1 and Black Oxide 2 are taken from Shapiro and Nuttle (1988). Compared with the computational results reported in that paper, the number of LPP's to be solved is much less (no more than 1/3 of the number reported in that paper). The problems Ligne 1 and Ligne 2 are taken from Manier-Lacoste (1994), the computation times of these two examples reported in that paper are 48 minutes and 53 seconds, and 1 minute and 45 seconds respectively. The average time for solving one LPP at the solution of the five problems is about 0.5 milliseconds. We also solve these LPP's by the simplex method, the average time for solving one is about 14.9 milliseconds. So, by taking advantages of both the branch and bound approach and the algorithm CTE developed in the paper, the total CPU times for solving these problems have been drastically reduced.

7. CONCLUSION

In this paper, we propose a branch and bound algorithm for generating an optimal cyclic schedule for hoist scheduling problems with time window constraints in PCB electroplating facilities. Furthermore, we propose a graph-based algorithm for solving the

LPP's. Computation results show that by taking advantages of both the branch and bound approach and the graph-based algorithm, the total CPU times for solving the problems are drastically reduced. The graph-based algorithm can also be used to solve any LPP which can be transformed into the cycle time evaluation of a bi-valued graph.

Further researches will be conducted to solve hoist scheduling problems with multiple product types and to attack hoist scheduling problems with a large number of tanks.

APPENDIX

Proof of the Correctness of Algorithm FLA:

Let $\lambda_i(k)$ denote the value of $\lambda(k)$ obtained at the end of loop 3.1 and $\Pi_i(k)$ the set of paths jointing vertex 1 to k and containing at most i backward arcs. We first claim that $\lambda_i(k) = \max_{\pi \in \Pi_i(k)} L(\pi)$. In the following, a path jointing vertex 1 to vertex k is referred to as 1->k path and each element in $\Pi_i(k)$ is considered as a string of arcs.

Algorithm FLA can be expressed as:

$$\Pi_i(k) = \max \{\lambda_{i-1}(k), \max_{a \in BI_k} \{\lambda_{i-1}(o(a)) + l(a)\}, \max_{a \in FI_k} \{\lambda_i(o(a)) + l(a)\}\} \qquad (5.4)$$

It is easy to see that:

$$\Pi_i(k) = \Pi_{i-1}(k) \cup \bigcup_{a \in BI(k)} \Pi_{i-1}(o(a)) \circ a \cup \bigcup_{a \in FI(k)} \Pi_i(o(a)) \circ a$$

where $\circ$ denotes the concatenation, so:

$$\max_{\pi \in \Pi_i(k)} L(\pi) = \max \{\lambda_{i-1}(k), \max_{a \in BI_k} \{\lambda_{i-1}(o(a)) + l(a)\}, \max_{a \in FI_k} \{\lambda_i(o(a)) + l(a)\}\}$$
$$= \lambda_i(k)$$

Now we can prove the correctness of the algorithm.

If function CHECK_CIRCUIT returns a value TRUE , then the graph has a circuit of positive length. Otherwise, the algorithm will terminate when $i > B'$.

Note that B' is the maximal number of backward arcs in the elementary circuits of the graph , so if the algorithm terminates when i>B', it means that all elementary circuits of the graph have been examined.

Therefore, the algorithm can correctly detect if there is a circuit of positive length in an ordinary graph.

It should be noted that before algorithm FLA terminates, none of the longest 1->k paths (k=1,2,...,n) found by the algorithm contains circuits.

Convergence of Algorithm CTE:

Since for any circuit γ^* of positive length belonging to G'_{LB}, $LB' = L(\gamma^*)/H(\gamma^*)$, we have $L'_{LB}(\gamma^*) = L(\gamma^*) - H(\gamma^*) \cdot LB' = 0$, i.e., γ^* is no longer a circuit of positive length of G'_{LB}, thus the convergence of algorithm CTE is the consequence of the fact that there is a finite number of elementary circuits in the graph.

Computation Complexity of Algorithm CTE:

Firstly, we show that the computation complexity of FLA is $O(n^2.m)$. At each execution of FLA, i takes at most n-1 values, and for each value of i, we perform an addition, a comparison, and at most one call of function CHECK_CIRCUIT for each of the m arcs of the graph. Since the computation complexity of function CHECK_CIRCUIT is O(n), thus the computation complexity of FLA is $O(n^2 \cdot m)$.

Secondly, we show that if the height of each arc of the bi-valued graph is 0, -1 or 1, then algorithm CTE calls FLA at most $n^2 \cdot m$ times.

Let $\Pi(i,k)$ denote the set of paths and circuits considered by FLA before and at iteration i, k. Then , $\Pi(i',k') \supseteq \Pi(i,k)$ for any $i' \geq i$, $k' \geq k$.

Suppose that at two successive calls of FLA in an execution of CTE, FLA finds a circuit γ of positive length of G'_{LB} at iteration i, k, and a circuit γ' of positive length of $G'_{LB'}$ at iteration i', k' respectively, then we have $i' \geq i$, $k' \geq k$, where $LB' = L(\gamma)/H(\gamma)$, $H(\gamma) > 0$, $H(\gamma') > 0$. If it is not true, then $\gamma' \in \Pi(i,k-1)$ if $k>1$, or $\gamma' \in \Pi(i-1,n))$ if $k=1$. Since $LB'>LB$, $L(\gamma') - H(\gamma') \cdot LB \geq L(\gamma') - H(\gamma') \cdot LB' > 0$ and γ' is a circuit of positive length of G'_{LB}, γ' should be found by FLA before iteration i, k.

Now, suppose that i'=i, k'= k. Then, since γ is one of the longest 1 -> k paths among all 1 -> k paths of G'_{LB} in $\Pi(i,k)$, we have

$$L(\gamma') - H(\gamma') \cdot LB \leq L(\gamma) - H(\gamma) \cdot LB$$

i.e., $$L(\gamma') - L(\gamma) - (H(\gamma') - H(\gamma)) \cdot LB \leq 0 \qquad (5.5)$$

From $L(\gamma') - H(\gamma') \cdot LB' > 0$ and $L(\gamma) - H(\gamma) \cdot LB' = 0$,

we obtain

$$L(\gamma') - L(\gamma) - (H(\gamma') - H(\gamma)) \cdot LB' > 0 \qquad (5.6)$$

From (5.5) and (5.6), we can derive

$$(H(\gamma') - H(\gamma)) \cdot (LB' - LB) < 0,$$

so $H(\gamma') < H(\gamma)$.

Since the height of any elementary circuit is no greater than n, the situation $i' \geq i$, $k' \geq k$ occurs at most m times. After that, either $i' > i$, or $i' = i$ and $k' > k$. So, algorithm CTE calls FLA at most $n^2 \cdot m$ times.

To sum up, the computation complexity of algorithm CTE in the worst case is $O(n^4 \cdot m^2)$.

ACKNOWLEDGMENTS

The authors thank Mr Pierre Fournet-Fayard for his help in computational experiment.

REFERENCES

Carlier, J. and Chrétienne, P. 1988, Les Problèmes d'Ordonnancement, Masson, Paris.

Hanen C. and Munier A. 1993, Ordonnancement cyclique d'un robot sur une ligne de galvanoplastie : modèles et algorithmes, Research Report LITP 93.30, Institut Blaise Pascal, Paris.

Lei L. and Wang T. J. 1989a, On the optimal cyclic schedules of single hoist electroplating processes, Working paper #89-0006, Rutgers University.

Lei L. and Wang T. J. 1989b, A proof: the cyclic hoist scheduling problem is NP-complete, Working paper #89-0016, Rutgers University.

Lei L. and Wang T. J. 1991, The minimun common-cycle algorithm for cyclic scheduling of two material handling hoists with the time window constraints, Management Science, 37, 12, 1629-1639.

Manier-Lacoste, M. A. 1994, Contribution à l'ordonnancement cyclique du système de manutention d'une ligne de galvanoplastie, Thèse de doctorat, Université de Franche-Comté.

Phillips, L. W. and Unger, P.S. 1976, Mathematical programming solution of a hoist scheduling problem, AIIE Transactions, 28, 2, 219-225.

Shapiro, G. W. and Nuttle, H. L. 1988, Hoist scheduling for a PCB electroplating facility, IIE Transactions, 20, 2, 157-167.

Unité de recherche INRIA Lorraine
Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - B.P. 101 - 54602 Villers lès Nancy Cedex (France)

Unité de recherche INRIA Rennes - IRISA, Campus universitaire de Beaulieu 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes - 46, avenue Félix Viallet - 38031 Grenoble Cedex 1 (France)
Unité de recherche INRIA Rocquencourt - Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis - 2004, route des Lucioles - B.P. 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 Le Chesnay Cedex (France)

ISSN 0249 - 6399