RC 16820 (#74431) 5/2/91
Manufacturing 34 pages

Research Report

An Approximate Algorithm for the Sequential Ordering Problem with Time Windows and Precedence Relationships

L. F. Escudero

IBM Research Division
T. J. Watson Research Center
Yorktown Heights, NY 10598

A. Sciomachen

Universita degli Studi di Milano
Milano, Italy

IBM Research Division
Almaden • T.J. Watson • Tokyo • Zurich

AN APPROXIMATE ALGORITHM FOR THE SEQUENTIAL ORDERING PROBLEM WITH TIME WINDOWS AND PRECEDENCE RELATIONSHIPS

L.F. Escudero

IBM Research, T.J. Watson Research Center, Yorktown Heights, NY

A. Sciomachen

Universita degli Studi di Milano, Italy

Abstract: Elsewhere we have introduced the sequential ordering problem with precedence relationships, SOP. In this paper we extent the problem to the case where time windows are considered. The problem has a broad range of applications, mainly, in production planning for manufacturing systems. Given a set of nodes, a weight associated with each node and a weight associated with each pair of nodes that can be sequenced consecutively, the SOP consists of finding a Hamiltonian path, such that release and due dates for each node and precedence relationships among the nodes are satisfied and a linear function is minimized. We present a tight formulation of the problem, and the framework for a cut generation LP based procedure for obtaining the optimal solution.

1. INTRODUCTION AND PROBLEM DEFINITION

The problems for flexible manufacturing systems we have in mind, see, e.g., [2], can be phrased in graph theoretical terminology in the following way. We are given a directed graph where an arc represents the possibility of performing two jobs consecutively and where a setup time is required for changing from one job to another. The following set of parameters are associated with each job: release date, due date and processing time. The process time is the number of time units that are necessary to process the job. The release date is the earliest time unit at which the processing can begin for the job. The due date is the time unit at which the job must be completed. In addition, some precedence relations are given that specify that some jobs have to be executed before certain others. A Hamiltonian path is the ordered sequence of all jobs. Let us define the makespan as the completion time of the job that is sequenced last in $\mathscr{H}$. The problem is to find a feasible $\mathscr{H}$ with minimum makespan, where a $\mathscr{H}$ is called *feasible* if it does not violate the release and due dates, nor the precedence constraints.

In this paper the given graph is the directed graph $G = (V,A)$ on n nodes (i.e., jobs). The following set of parameters is associated with each node $i \in V$ p_i, processing time; r_i, release date; d_i, due date. We denote an arc going from some node i to another node j by (i,j) and the associated setup time by c_{ij}. (It is assumed that $(i,i) \notin A$.)

Let $\mathscr{H}$ denote a Hamiltonian path through the set of nodes V. Let $i \rightarrow j$ mean that node i is the immediate predecessor of node j in the given $\mathscr{H}$. Let $sp(i \rightarrow j)$ denote the ordered set of nodes in the successor path from node i to node j in $\mathscr{H}$.

Let t_i denote the accumulated potential of node i (i.e., the completion time of job i) in a given $\mathscr{H}$ in G. Note that t_i depends on the set of nodes sequenced before node i in $\mathscr{H}$. Then it can be expressed as

$$t_i = \max \{r_i, t_{y(i)} + + c_{y(i),i}\} + p_i \tag{1.1}$$

where $y(i)$ denotes the node that is the immediate predecessor of node i in $\mathscr{H}$. (Note. $i \rightarrow j$ means that $y(j) = i$.)

For simplicity let 0 be a dummy node such that $p_0 = 0$, $r_0 = 0$, $d_0 = 0$, $c_{i,0} = 0$ and $c_{0,i} = s_i$ for $i \in V$, where s_i gives the weight (i.e., setup time) of node i provided that it is the head node of the given $\mathcal{H}$.

The precedence constraints are given by a digraph $\mathcal{I} = (N,P)$ where $N \subseteq V$ and an arc $(i,j) \in P$ means that job i has to be performed before job j. Clearly, this *precedence digraph* $\mathcal{I}$ has to be acyclic (i.e., may not contain a directed cycle). Moreover, if $(i,j),(j,k) \in P$ then k cannot be performed before i, in other words, we can also assume that $\mathcal{I}$ is transitively closed. So the precedence constraints are given by an acyclic and transitively closed digraph $\mathcal{I} = (N,P)$. Note that $(i,j) \in P$ means that $t_i + S_{i,j} \leq t_j$, where $S_{i,j}$ is the distance of the shortest path from node i to node j in digraph $G = (V,A)$, where the distance from some node a to node b is $T_{a,b}$ for $(a,b) \in A$. ($T_{a,b} \equiv c_{a,b} + p_b$.) Let $(0,b) \in P$ for all $b \in V$. So now $\mathcal{I} \neq \emptyset$ and $V := V \bigcup 0$ and $A := A \bigcup B$, where the arc set B is given by the expression

$$B = \{(0,b)\} \text{ where } b \in V \text{ such that } T_{0,b} \leq d_b \text{ and } \nexists a \in V | (a,b) \in P \tag{1.2}$$

Let us call a Hamiltonian path in G *feasible* if the two following conditions are satisfied

$$r_i - 1 + p_i \leq t_i \leq d_i \tag{1.3}$$

and

$$(j,i) \notin P \text{ holds for all } i \prec j \tag{1.4}$$

where $i \prec j$ means that there is a successor path from node i to node j in $\mathcal{H}$.

Now we can state the *Sequential Ordering Problem (SOP)* [2] formally. Given a digraph $G = (V,A)$ with setup time c_{ij} for all $(i,j) \in A$, the parameters p_i, r_i and d_i for node $i \in V$ and a transitively closed acyclic digraph $\mathcal{I} = (N,P)$, find a feasible Hamiltonian path $\mathcal{H}$ in G that has minimum accumulated potential (i.e., completion time) for the node sequenced last.

If the precedence digraph $\mathcal{I} = (N,P)$ has empty arc set and no release and due dates are required, the SOP reduces to finding a minimum cost Hamiltonian path in G.

This is an *NP*-hard problem and so is the SOP. Our main concern here, though, is not an algorithm for the "pure" Hamiltonian path problem (or, equivalently, the asymmetric traveling salesman problem, ATSP) but a method that deals with precedences, release and due dates.

Since we are dealing with large-scale cases, we are interested in fast approximate algorithms. This paper presents procedures for obtaining good feasible solutions. It is organized as follows. Section 2 presents our 0-1 model for the SOP. A preprocessing procedure is presented in Section 3. Section 4 describes our procedure for getting a feasible solution to the problem. Sections 5 and 6 present the basic ideas of local search procedures for improving feasible solutions. Section 7 gives an illustrative example. Finally, we offer some conclusions and outline future work.

2. The 0-1 MODEL

Let $x_{i,j}$ be a 0-1 variable such that $x_{i,j} = 1$ means that $i \rightarrow j$, otherwise, it is zero. Let t_i be a continuous variable that gives as above the *ap* of node i (i.e., the completion time of job i).

There are several equivalent 0-1 models for the SOP. Our proposed model is as follows

$$min \ \{t^* \mid t_v \leq t^* \ \forall v \in V\} \tag{2.1}$$

subject to

$$x(A) = n - 1 \tag{2.2}$$

$$x(\delta^-(v)) = 1 \text{ for all } v \in V \setminus v \neq 0 \tag{2.3}$$

$$x(\delta^+(a)) = 1 \text{ for all } a \in V \text{ such that } \exists b \in V \setminus (a,b) \in P \tag{2.4a}$$

$$x(\delta^+(a)) \leq 1 \text{ for all } a \in V \text{ such that } \nexists b \in V \setminus (a,b) \in P \tag{2.4b}$$

$$0 \leq x_{ij} \leq 1 \text{ for all } (i,j) \in A \tag{2.5}$$

$$x(A(W)) \leq |W| - 1 \text{ for all } W \subset V, 2 \leq |W| \leq n - 1 \tag{2.6}$$

$$x_{i,j} \in \{0;1\} \text{ for all } (i,j) \in A \tag{2.7}$$

$$m_{i,j}(1 - x_{i,j}) \le t_i + T_{i,j} - t_j \le M_{i,j}(1 - x_{i,j}) \quad \forall (i,j) \in A \text{ for } r_v = 1 \quad \forall v \in V \tag{2.8a}$$

$$t_i + T_{i,j} - t_j \le M_{i,j}(1 - x_{i,j}) \quad \forall (i,j) \in A \text{ for } \exists v \in V \mid r_v > 1 \tag{2.8b}$$

$$t_a + S_{a,b} \le t_b \quad \forall (a,b) \in P \tag{2.9}$$

$$e_v \le t_v \le d_v \text{ for all } v \in V \tag{2.10}$$

where $x(F) = \sum_{(i,j) \in F} x_{i,j}$, $\delta^-(v) = \{(w,v) \in A\}$, $\delta^+(v) = \{(v,w) \in A\}$, $A(W) = \{(i,j) \in A | i,j \in W\}$, $e_i = r_i - 1 + p_i$. and

$$\begin{aligned} M_{i,j} &= (d_i + T_{i,j}) - e_j \\ m_{i,j} &= (e_i + T_{i,j}) - d_j \end{aligned} \tag{2.11}$$

Let γ be a 0-1 parameter, such that $\gamma = 0$ for $r_v = 1 \ \forall v \in V$ and, otherwise, $\gamma = 1$.

The minimum makespan is defined in (2.1). Constraint (2.2) defines the number of arcs in any $\mathcal{H}$. Constraint (2.3) for node $v \in V | v \neq 0$ (resp., (2.4a) for any node that cannot be the last node in any feasible $\mathcal{H}$) forces a predecessor path (resp., successor path) starting with the given node. Constraints (2.4b) prevent that more than one node is sequenced immediately after any other node that can be the last node in any feasible $\mathcal{H}$. Constraints (2.6) are the so-called *Subtour Elimination Constraints* SECs (see e.g. Ascheuer et al. [1]). Constraint (2.8a) forces $t_j = t_i + T_{i,j}$ whenever $x_{i,j} = 1$ for $\gamma = 0$ and any $(i,j) \in A$; otherwise, it is not active. Note that (2.8b) together with (2.10) forces $max\{r_j - 1, t_i + c_{i,j}\} + p_j \le t_j \le d_j$ whenever $x_{i,j} = 1$ for $\gamma = 1$ and $(i,j) \in A$. Constraints (2.9) are the so-called *Precedence Forcing Constraints* PFCs such that node a must be sequenced before node b for $(a,b) \in P$. (Note that the PFC (2.9) for $(a,b) \in P$ cannot be satisfied if $e_a + S_{a,b} > d_b$.) See in [1] an equivalent formulation for enforcing the precedence relationships given by P in a non-release/due date environment. The ap of each node is bounded by (2.10). The big enough values of $M_{i,j}$ and $m_{i,j}$ given in (2.11) do not cut-off any integer solution but are hopefully tight enough for the LP relaxation problem. Let (2.8) together with (2.10) be named *release and due Date*

Forcing Constraints DFCs. (Note that constraint (2.8a) is LP tighter than constraint (2.8b)).

In a companion paper (see [4]) we outline a cut generation LP procedure for obtaining the optimal solution and, in any case, a tight lower bound for model (2.1)-(2.10).

3. PREPROCESSING

3.1. TIGHTENING THE ARC SET A

Note that arc (i,j) should be deleted from set A provided that it has been detected that $i \not\rightarrow j$ in any feasible $\mathscr{H}$.

A rough procedure is as follows.

1. Fix $x_{ij} = 0$ (i.e., delete arc (i,j) from set A) provided that any of the following conditions is satisfied.

$$(j,i) \in P \tag{3.1}$$

$$(i,k),(k,j) \in P \tag{3.2}$$

$$e_i + T_{ij} > d_j \tag{3.3}$$

$$\exists k \in V \text{ such that either } x_{i,k} = 1 \text{ or } x_{kj} = 1 \tag{3.4}$$

2. Fix $x_{ij} = 1$ provided that any of the following conditions is satisfied.

$$x_{kj} = 0 \quad \forall k \in V \text{ such that } k \neq i \text{ and } \exists \ell | (\ell, j) \in P \quad \forall \ell \in V \tag{3.5}$$

$$x_{i,k} = 0 \quad \forall k \in V \text{ such that } k \neq j \text{ and } \exists \ell | (i, \ell) \in P \quad \forall \ell \in V \tag{3.6}$$

Condition (3.1) is implied by constraint (2.9). Condition (3.2) anticipates $i \not\rightarrow j$ in any feasible $\mathscr{H}$. (Note that $(i,j) \in P$ since P is a transitively closed arc set.) Observe that condition (3.3) implies $i \not\rightarrow j$ in any feasible $\mathscr{H}$. Condition (3.4) is implied by constraints (2.3) and (2.4). Condition (3.5) (resp., condition (3.6)) implies $i \rightarrow j$ in any feasible $\mathscr{H}$; notice that it states that node j (resp., node i) cannot be the initial (resp., final) node of any feasible $\mathscr{H}$ and all other potential relationships $k \rightarrow j$ (resp., $i \rightarrow k$) are not allowed.

3.2. REDUCING THE DUE/RELEASE DATES GAP

Notice that smaller gap $d_v - r_v$, $v \in V$, tighter DFCs constraints (2.8) and (2.10). In any case, the procedure described in this section is only effective for $P \neq \emptyset$.

The procedure below tightens the release and due dates based on the two following simple observations.

1. The processing of a given job cannot begin before the completion of its predecessor nodes in P, nor before the completion of the node to be sequenced immediately before it in $\mathscr{H}$ plus its related setup time.
2. The completion of a given job cannot be performed after beginning the processing of its sucessor nodes in P, nor after beginning the processing of the node to be sequenced immediately after it in $\mathscr{H}$ minus its related setup time.

In a recursive mode the vectors r and d should be updated as follows.

Tightening r_j all $j \in V$

$$r_j := \max \{r_j , R1 , R2\} \tag{3.7}$$

where

$$R1 \equiv \max \{r_i + p_i + \underline{C}_i \quad \forall i|(i,j) \in P\} \tag{3.8}$$

$$R2 \equiv \underline{R}^j + \left(\sum_{i|(i,j) \in P} p_i + \underline{C}_i \right) \tag{3.9}$$

and

$$\underline{C}_i = \min \{c_{i,k} \quad \forall k|(i,k) \in A\} \tag{3.10}$$

$$\underline{R}^j = \min \{r_i \quad \forall i|(i,j) \in P\} \tag{3.11}$$

Tightening d_i all $i \in V$

$$d_i := \max \{d_i , D1 , D2\} \tag{3.12}$$

where

$$D1 \equiv \min \{d_j - (p_j + \underline{C}^j) \quad \forall j|(i,j) \in P\} \tag{3.13}$$

$$D2 \equiv \underline{D}_i - \left(\sum_{j|(i,j) \in P} p_j + \underline{C}^j \right) \tag{3.14}$$

and

$$\underline{C}^j = \min \{c_{kj} \quad \forall k | (k,j) \in A\} \tag{3.15}$$

$$\underline{D}_i = \max \{d_j \quad \forall j | (i,j) \in P\} \tag{3.16}$$

Note. $1 \leq r_i$ and $d_j \leq \bar{z}$, where $\bar{z}$ denotes the upper bound of the *ap* of the node to be sequenced last.

For illustrative purposes let us consider the case shown in Table 1.

Table 1. Sample case for reducing release/due date gap

node	dates		weight
	release	due	
1	1	20	5
2	35	50	8
3	1	55	5
4	40	75	15
5	40	89	17

Assumptions: $(1,3),(2,3)(3,4),(3,5) \in P$

$$c_{1,3} = c_{2,3} = 10$$

It is easy to see that our procedure reduces the gap $\{d_i - r_j\}$ as shown in Table 2. Observe that there is not flexibility on scheduling the nodes, but node 1. In fact, $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5$ is the unique feasible sequence.

Table 2. New release/due dates for cases shown in Table 1

node	dates	
	release	due
1	1	20
2	35	47
3	53	57
4	58	75
5	58	89

3.3. TIGHTENING THE ARC SET P

Delete (a,b) from P provided that the following condition is satisfied

$$d_a - (\underline{C}^a + p_a) < r_b + p_b \tag{3.17}$$

where

$$\underline{C}^a = \min \{c_{i,a} \quad \forall i | (i,a) \in A\} \tag{3.18}$$

Note that (2.9) is redundant whenever condition (3.17) is satisfied.

We suggest to execute sequentially the procedures 3.1, 3.2 and 3.3. The sequence should be repeated till not further changes are obtained.

3.4. DETECTING INFEASIBILITY

A given instance is infeasible if any of the following conditions is satisfied, at least

1. $B = \emptyset$, where B is given by (1.2).
2. $\exists v \in V | e_v > d_v$.
3. $\exists (a,b) \in P$ and $\nexists (a,j) \in A$
4. $|L^A| > 1$

5. $\exists(a,b) \in P$ and $\nexists(i,b) \in A$

6. $|L^t| > 1$

7. $a \in L^h$ and $a \notin B$

8. $\exists a,b \in V \mid (a,b),(b,a) \in P$.

9. $\exists a,b,c \in V \mid (a,b),(b,c),(c,a) \in P$.

10. $G = (V, A)$ is not a connected graph.

where L^h (resp., L^t) is the set of *forced* nodes at the head (resp., tail) of any feasible $\mathscr{H}$. Then,

$$L^h = \{a \mid \exists(a,b) \in P \text{ and } \nexists(i,a) \in A\} \tag{3.19}$$

$$L^t = \{b \mid \exists(a,b) \in P \text{ and } \nexists(b,j) \in A\} \tag{3.20}$$

Condition 1 states that no node can be the head node of any feasible $\mathscr{H}$. Condition 2 states that a due date cannot be satisfied. Condition 3 states that there is a node, say a that must be the predecessor of some other node but, on the other hand, it must be the tail node in any feasible $\mathscr{H}$. Similarly, condition 5 states that there is a node, say b that must be the successor of some other node but, on the other hand, it must be the head node in any feasible $\mathscr{H}$. Conditions 4 (resp., condition 6) states that more than one node is forced to be the head node (resp., the tail node) of any $\mathscr{H}$. Condition 7 states that a node, say a is forced to be the head node but, on the other hand, $0 \nrightarrow a$ in any feasible $\mathscr{H}$. Conditions 8 and 9 detect 2- and 3-cycles in P, respectively.

4. OBTAINING A FEASIBLE SOLUTION

4.1. FRAMEWORK

For simplicity in our exposition let us assume that $T_{v,0} = 0$ for all $v \in V$ and, then, $\mathscr{H}$ can also be viewed as a *circuit*. Also let $T_{e,h} = \infty$ for $(a,b) \notin A$.

Assume that we have obtained a Hamiltonian circuit $\mathscr{H}$ either by using general algorithms that do not take into account the nature of the SOP (see [8] among others), or by using ad-hoc algorithms that consider some of the features of the problem. Elsewhere ([5]) we present an algorithm that produces a Hamiltonian path $\mathscr{H}$ based on the EDD rule (*earliest due date*). Let $x(v)$ give the node such that $v \rightarrow x(v)$ belongs to $\mathscr{H}$; let also $y(x(v)) = v$. Then, the Hamiltonian path is given by $\mathscr{H} = sp(x(0) \rightarrow y(0))$. (I.e., $\mathscr{H} = x(0) \rightarrow x(x(0))...y(y(0)) \rightarrow y(0)$.)

Let $t(v)$ denote the *ap* of node v in $\mathscr{H}$. It can be expressed

$$t(v) = max\ \{r_v - 1\ ,\ t(y(v)) + c_{y(v),v}\} + p_v \tag{4.1}$$

So, the release date constraints are satisfied. Note that $x(v)$ and $t(v)$ are denoted by $x_{i,x(i)} = 1$ and t_i in model (2.1)-(2.10).

In this section we describe a methodology for modifying the initial $\mathscr{H}$ such that the DFCs and PFCs (2.8)-(2.10) are satisfied. The procedure performs two *restricted* TRIA local searches and one (*reverse*) BI local search that hopefully reduce $t(y(0))$, besides providing a feasible solution. Let $a \prec b$ mean that node a is ordered (immediately or not) before node b in any $\mathscr{H}$.

The *restricted* TRIA-1 local search works on the nodes pair (i,j) such that the conditions

$$\begin{aligned} &t(v) \le d_v \quad \forall v \in sp(x(0) \rightarrow y(j)) \quad \text{and} \\ &\nexists b,a \in sp(x(0) \rightarrow y(j)) | b \prec a \ \text{ such that } (a,b) \in P \end{aligned} \tag{4.2}$$

$$t(j) > d_j \quad \text{or} \quad \exists b \in sp(x(0) \rightarrow y(j)) \ \text{ such that } (j,b) \in P \tag{4.3}$$

are satisfied by the sequence

$$sp(0 \rightarrow i) \rightarrow sp(x(i) \rightarrow y(j)) \rightarrow j \rightarrow sp(x(j) \rightarrow y(0)) \tag{4.4}$$

The sequence (4.4) will be replaced (see Figure 1) by the sequence

$$sp(0 \rightarrow i) \rightarrow j \rightarrow sp(x(i) \rightarrow y(j)) \rightarrow sp(x(j) \rightarrow y(0)) \tag{4.5}$$

provided that the new $\mathscr{H}$ satisfies conditions (4.6)-(4.7).

$$\tilde{t}(v) \le d_v \quad \forall v \in j \rightarrow sp(x(i) \rightarrow y(j)) \tag{4.6}$$

$$\nexists a \in sp(x(i) \rightarrow y(j)) \ \text{ such that } (a,i) \in P \tag{4.7}$$

where $\bar{i}(v)$ gives the new *ap* of node v. (See Sections 4.2 and 4.3.) (Note. $i = 0$ and $j = y(0)$ are allowed.)

Let us define the range $[\kappa_j, K_j]$ of node i for which the new partial sequence (4.5) can be feasible. Let κ_j and K_j be as follows

$$\kappa_j\text{: last node from } sp(0 \to y(y(j))) \text{ such that } (\kappa_j, j) \in P \tag{4.8}$$

(Note that by construction $\kappa_j = 0$ is allowed. It is the case where $\nexists a \in sp(0 \to y(y(j)))|(a,j) \in P$.)

$$K_j: \begin{cases} \text{first node from } sp(0 \to y(y(j))) \text{ such that } (j, x(K_j)) \in P, \text{ or} \\ y(y(j)) \text{ such that } \nexists b \in sp(x(0) \to y(j))|(j,b) \in P \end{cases} \tag{4.9}$$

Note that any node i such that either $K_j \prec i$ for $(j, x(K_j)) \in P$, or $i \prec \kappa_j$ for $(\kappa_j, j) \in P$ violates condition (4.7). Note that $K_j \prec \kappa_j$ means that there is not any sequence (4.5) that satisfies (4.7) for the given node j.

(When no precedence relations are present, let us slightly abuse the meaning of the arc set P, such that we will assume in this case that $\nexists k \in V$ such that $(k,j) \in P$, $(j,k) \in P$. Then, κ_j and K_j can also be obtained by using (4.8)-(4.9).)

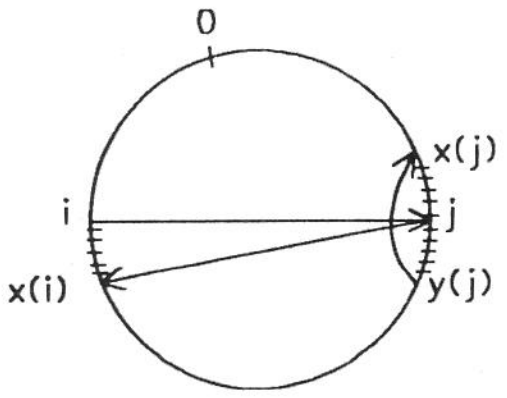

Figure 1. A restricted TRIA-1 change searching for DFCs-PFCs feasibility

The *restricted* TRIA-2 local search works on the nodes pair (i,j) such that conditions (4.2)-(4.3) are satisfied by the sequence given by (4.4). The sequence (4.4) will be replaced (see Figure 2) by the sequence

$$sp(0 \rightarrow y(i)) \rightarrow sp(x(i) \rightarrow j) \rightarrow i \rightarrow sp(x(j) \rightarrow y(0)) \tag{4.10}$$

provided that the new $\mathcal{H}$ satisfies conditions (4.11)-(4.12).

$$\bar{t}(j) \leq d_j \text{ and } \bar{t}(i) \leq d_i \tag{4.11}$$

$$\nexists b \in sp(x(i) \rightarrow j) \text{ such that } (i,b) \in P \tag{4.12}$$

where $\bar{t}(v)$ gives the new *ap* of node v. (See Sections 4.2 and 4.3.) (Note. $i = 0$ and $j = y(0)$ are allowed.)

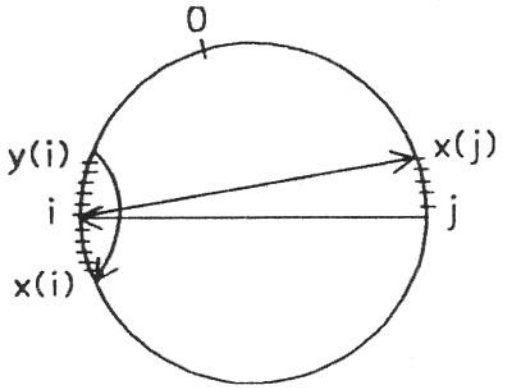

Figure 2. A restricted TRIA-2 change searching for DFCs PFCs feasibility

Let $pp(j \rightarrow x(i))$ denote the ordered set of nodes in the *predecessor path* from node j to node $x(i)$ in the current $\mathcal{H}$. (I.e., $j, y(j), \ldots, x(x(i)), x(i)$.) The (*reverse*) BI local search works also on the pair (i,j) such that conditions (4.2)-(4.3) are satisfied by the sequence given by (4.4)). The sequence (4.4) will be replaced (see Figure 3) by the sequence

$$sp(0 \rightarrow i) \rightarrow pp(j \rightarrow x(i)) \rightarrow sp(x(j) \rightarrow y(0)) \tag{4.13}$$

provided that the new $\mathcal{H}$ satisfies conditions (4.14)-(4.15).

$$\bar{t}(v) \leq d_v \qquad \forall v \in pp(j \rightarrow x(i)) \tag{4.14}$$

$\nexists a,b \in V$ such that $b \in pp(j \rightarrow x(x(i)))$, $a \in pp(y(b) \rightarrow x(i))$ and $(a,b) \in P$ (4.15)

where $\bar{t}(v)$ gives the new *ap* of node v. (See Sections 4.2 and 4.3.) (Note. $i = 0$ and $j = y(0)$ are allowed. Note also that (*reverse*) BI allows for the full reverse sequence $0 \rightarrow pp(j \rightarrow x(0))$.)

Let κ_j be expressed as follows

κ_j : last node from $pp(y(y(j)) \rightarrow 0)$ (4.16)

such that conditions (4.14)-(4.15) are satisfied, where i is replaced by κ_j. Note that any node i such that $i \prec \kappa_j$ violates condition (4.15).

(When no precedence relations are present, let us again slightly abuse the meaning of the arc set P, such that we will assume in this case that $\nexists a,b \in V$ such that $(a,b) \in P$. Then, κ_j can also be obtained by using (4.16).)

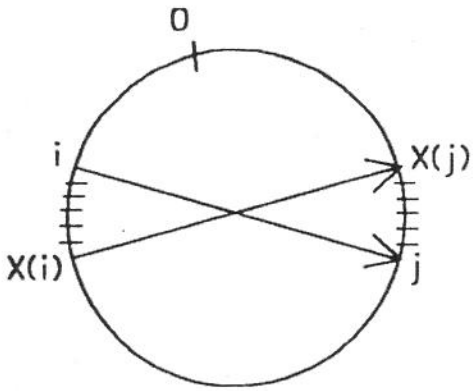

Figure 3. A (reverse) BI change searching for DFCs/PFCs feasibility

A common feature of the three above procedures is the mechanism that forces the infeasible node at a current major iteration to be included in the appropriate nodes' interchange. It speeds up the procedures, but may fail to find feasible solutions. Let the case given in Tables 3 and 4. Assume zero setup time. The EDD-based initial solution is $1 \rightarrow 2 \rightarrow 4 \rightarrow 3 \rightarrow 6 \rightarrow 5$. Note that the DFC for node #6 is violated ($t_6 = 35$, $d_6 = 34$).

No *restricted* TRIA-1, *restricted* TRIA-2 and *reverse* BI-based interchanges produce a feasible solution. Nevertheless, the procedure TRIA, see Section 5, finds the feasible sequence. We recommend to use it only after detecting the failure of the other procedures, since it may require more computational effort.

Table 3. Case where a feasible solution is not found

node #	dates		weight
	release	due	
1	1	9	5
2	10	20	5
3	15	45	10
4	6	19	10
5	25	60	15
6	16	34	15

Table 4. Precedence set P = {(a,b)} for case given in Table 3

a	b
1	2,3,4,5,6
2	3,5
3	5
4	5,6

4.2. NO RELEASE DATES

Here below we give the expressions for the nodes' *ap* $\{\bar{t}(v)\}$ related to the new partial sequence when no release dates are required, i.e., $r_v = 1$ for all $v \in V$. Note that, in this case, $P = \emptyset$ (no precedence relations are considered); otherwise, our preprocessing procedure would update the release dates appropriately (see Section 3.2).

Let $\alpha_{a,b}$ denote the *gain* obtained by *inserting* node b in the partial sequence $sp(0 \rightarrow x(a))$, such that the new sequence is $sp(0 \rightarrow a) \rightarrow b \rightarrow x(a)$. It can be expressed

$$\alpha_{a,b} = T_{a,x(a)} - (T_{a,b} + T_{b,x(a)}) \tag{4.17}$$

Let β_b denote the *gain* obtained by *deleting* node b from the partial sequence $sp(0 \rightarrow x(b))$, such that the new sequence is $sp(0 \rightarrow y(b)) \rightarrow x(b)$. It can be expressed

$$\beta_b = T_{y(b),b} + T_{b,x(b)} - T_{y(b),x(b)} \tag{4.18}$$

Restricted TRIA-1 local search (partial sequence (4.5) up to $y(j)$)

$$\bar{t}(j) = t(i) + T_{ij} \tag{4.19}$$

$$\bar{t}(v) = t(v) - \alpha_{ij} \qquad \forall v \in sp(x(i) \rightarrow y(j)) \tag{4.20}$$

Restricted TRIA-2 local search (partial sequence (4.10) up to i)

$$\bar{t}(v) = t(v) - \beta_i \qquad \forall v \in sp(x(i) \rightarrow j) \tag{4.21}$$

$$\bar{t}(i) = \bar{t}(j) + T_{ji} \tag{4.22}$$

(Reverse) BI local search (partial sequence (4.13) up to $x(i)$)

$$\bar{t}(j) = t(i) + T_{ij} \tag{4.23}$$

$$\bar{t}(v) = \bar{t}(x(v)) + T_{x(v),v} \qquad \forall v \in pp(y(j) \rightarrow x(i)) \tag{4.24}$$

4.3. RELEASE DATES

Here below we give the expressions for the nodes' *ap* $\{\bar{t}(v)\}$ related to the new partial sequence when release dates are present, i.e., $\exists v \in V | r_v > 1$. (By construction, the initial solution provided by (4.1) *does* satisfy the release date constraints. The methodology for obtaining a feasible solution to be described below will always preserve this type of constraints at any step of the procedure.)

For the *inserting* case, $\alpha_{a,b}$ can be expressed as follows

$$\alpha_{a,b} = t(x(a)) - \bar{t}(x(a)) \tag{4.25}$$

where

$$\bar{t}(x(a)) = max\ \{r_{x(a)} - 1\ ,\ \bar{t}(b) + c_{b,x(a)}\}\ +\ p_{x(a)} \tag{4.26}$$

and

$$\bar{t}(b) = max\ \{r_b - 1\ ,\ t(a) + c_{a,b}\}\ +\ p_b \tag{4.27}$$

For the *deleting* case, β_b can be expressed as follows

$$\beta_b = t(x(b)) - \bar{t}(x(b)) \tag{4.28}$$

where

$$\bar{t}(x(b)) = max\ \{r_{x(b)} - 1\ ,\ t(y(b)) + c_{y(b),x(b)}\}\ +\ p_{x(b)} \tag{4.29}$$

The expressions for the nodes' ap are as follows.

Restricted TRIA-1 local search (partial sequence (4.5) up to $y(j)$)

$$\bar{t}(j) = max\ \{r_j - 1\ ,\ t(i) + c_{i,j}\}\ +\ p_j \tag{4.30}$$

$$\bar{t}(x(i)) = max\ \{r_{x(i)} - 1\ ,\ \bar{t}(j) + c_{j,x(i)}\}\ +\ p_{x(i)} \tag{4.31}$$

$$\bar{t}(v) = max\ \{r_v - 1\ ,\ \bar{t}(y(v)) + c_{y(v),v}\}\ +\ p_v \qquad v \in sp(x(x(i)) \rightarrow y(j)) \tag{4.32}$$

Restricted TRIA-2 local search (partial sequence (4.10) up to i)

$$\bar{t}(x(i)) = max\ \{r_{x(i)} - 1\ ,\ t(y(i)) + c_{y(i),x(i)}\}\ +\ p_{x(i)} \tag{4.33}$$

$$\bar{t}(v) = max\ \{r_v - 1\ ,\ \bar{t}(y(v)) + c_{y(v),v}\}\ +\ p_v \qquad \forall v \in sp(x(x(i)) \rightarrow j) \tag{4.34}$$

$$\bar{t}(i) = max\ \{r_i - 1\ ,\ \bar{t}(j) + c_{j,i}\}\ +\ p_i \tag{4.35}$$

(Reverse) BI local search (partial sequence (4.13) up to $x(i)$)

$$\bar{t}(j) = max\ \{r_j - 1\ ,\ t(i) + c_{i,j}\}\ +\ p_j \tag{4.36}$$

$$\bar{t}(v) = max\ \{r_v - 1\ ,\ \bar{t}(x(v)) + c_{x(v),v}\}\ +\ p_v \qquad \forall v \in pp(y(j) \rightarrow x(i)) \tag{4.37}$$

4.4. ROUGH ALGORITHM

Let $\gamma = 0$ denote the case where no release dates are required (i.e., $r_v = 1$ for all $v \in V$) and, otherwise, $\gamma = 1$ (i.e., $\exists v \in V | r_v > 1$).

Formally, the procedure is as follows.

For $j \in sp(x(0) \rightarrow y(0))$ such that condition (4.3) is satisfied for $t(j)$ as given by (4.1).

Set $\Delta := -\infty$.

Execute *restricted* TRIA-1.

If $\Delta > -\infty$ then obtain the new incumbent $\mathscr{H}$ (4.5) and update vector t

by using the pair $(\bar{i}, j)$ in (4.19)-(4.20) for $\gamma = 0$

and (4.30)-(4.32) for $\gamma = 1$

Otherwise, execute *restricted* TRIA-2.

If $\Delta > -\infty$ then obtain the new incumbent $\mathscr{H}$ (4.10) and update vector t

by using the pair $(\bar{i}, j)$ in (4.21)-(4.22) for $\gamma = 0$

and (4.33)-(4.35) for $\gamma = 1$

Otherwise, execute (*reverse*) BI.

If $\Delta > -\infty$ then obtain the new incumbent $\mathscr{H}$ (4.13) and update vector t

by using the pair $(\bar{i}, j)$ in (4.23)-(4.24) for $\gamma = 0$

and (4.36)-(4.37) for $\gamma = 1$

Otherwise, stop. (No feasible solution is found.)

End

The *restricted* TRIA-1 is as follows.

For $i \in sp(\kappa_j \rightarrow K_j)$ such that condition (4.6) is satisfied where

κ_j and K_j as given by (4.8)-(4.9), and

$\bar{t}(v)$ as given by (4.19)-(4.20) for $\gamma = 0$ and (4.30)-(4.32) for $\gamma = 1$

Save $\bar{i} := i$ and update Δ such that

$$\Delta := \begin{cases} max\ \{\Delta,(\alpha_{ij} + \beta_j)\} & \text{for } \gamma = 0 \\ max\ \{\Delta,(t(j) - \bar{t}(y(j)))\} & \text{for } \gamma = 1 \end{cases}$$

End

Note that condition (4.6) is not required to be tested for any node $v \in sp(x(i) \rightarrow y(j))$, such that $\alpha_{ij} \geq 0$. See (4.20) for $\gamma = 0$ and (4.25) for $\gamma = 1$.

The *restricted* TRIA-2 is as follows.

For $i \in sp(x(0) \rightarrow y(j))$ such that conditions (4.11)-(4.12) are satisfied where

$\bar{t}(v)$ as given by (4.21)-(4.22) for $\gamma = 0$ and (4.33)-(4.35) for $\gamma = 1$

Save $\bar{i} := i$ and update Δ such that

$$\Delta := \begin{cases} max\ \{\Delta,(\beta_i + \alpha_{jj})\} & \text{for } \gamma = 0 \\ max\ \{\Delta,(t(j) - \bar{t}(i))\} & \text{for } \gamma = 1 \end{cases}$$

End

Note that condition (4.11) is not required to be tested for any node $v \in sp(x(i) \rightarrow j)$, such that $\beta_i \geq 0$. See (4.21) for $\gamma = 0$ and (4.28) for $\gamma = 1$.

The (*reverse*) BI is as follows.

For $i \in sp(\kappa_j \rightarrow y(y(j)))$ where

κ_j as given by (4.16), and

$\bar{t}(v)$ as given by (4.23)-(4.24) for $\gamma = 0$ and (4.36)-(4.37) for $\gamma = 1$

Save $\bar{i} := i$ and update Δ such that

$$\Delta := \begin{cases} max\ \{\Delta,\ t(x(j)) - \bar{t}(x(j))\} & \text{for } \gamma = 0 \\ max\ \{\Delta,(t(j) - \bar{t}(x(i)))\} & \text{for } \gamma = 1 \end{cases}$$

End

The discussion of the implementation's details of the algorithm is beyond the scope of this paper. The current version given above is intended only for expository purposes. For example, smaller computational effort may require the following version of *(reverse)* BI

For $i \in pp(y(y(j)) \rightarrow 0)$ until either $\exists \ell \in pp(j \rightarrow x(x(i)) | (x(i), \ell) \in P$ or $\bar{t}(x(i)) > d_{x(i)}$

Save $\bar{i} := i$ and update Δ as above

End

See [5] for a thorough description of the implementation.

5. A TRIA LOCAL SEARCH FOR IMPROVING FEASIBLE SOLUTIONS

5.1. FRAMEWORK

This section describes a TRIA local search for improving the initial feasible $\mathcal{H}$. (See related work in Kanellakis and Papadimitrou [7] for the ATSP and Escudero [2,3] for the SOP without time windows.) We perform a local search for obtaining a Hamiltonian circuit with a better makespan, while satisfying the DFCs and PFCs. TRIA works on the nodes tuple (i,j,k), such that the new partial orderings (see Figure 4) are $\bar{x}(i) := j$, $\bar{x}(y(j)) := k$ and $\bar{x}(y(k)) := x(i)$, where $\bar{x}(v)$ gives the successor node of node v in the new sequence.

The TRIA search seeks to improve the makespan $t(y(0))$ of the current $\mathcal{H}$

$$sp(0 \rightarrow i) \rightarrow sp(x(i) \rightarrow y(j)) \rightarrow sp(j \rightarrow y(k)) \rightarrow sp(k \rightarrow y(0)) \tag{5.1}$$

by finding the tuple (i,j,k) such that the new sequence

$$sp(0 \rightarrow i) \rightarrow sp(j \rightarrow y(k)) \rightarrow sp(x(i) \rightarrow y(j)) \rightarrow sp(k \rightarrow y(0)) \tag{5.2}$$

gives a better makespan, i.e., $\bar{t}(y(0)) < t(y(0))$ and satisfies the DFCs (5.3) and PFCs (5.4).

$$\bar{t}(v) \leq d_v \quad \text{for all } v \in V \tag{5.3}$$

$$a \prec b \quad \text{for all } (a,b) \in P \tag{5.4}$$

where $\bar{t}(v)$ gives the new *ap* of node v (see below).

A node, say i is termed an *admissible* one if there is not evidence that it cannot be selected as the first one in the nodes' tuple. Formally, node i is an admissible one whenever condition (5.5) is satisfied

$$c_{i,x(i)} \geq \varepsilon_1 \quad \text{or} \quad r_{x(i)} - 1 - (t(i) + c_{i,x(i)}) \geq \varepsilon_2 \tag{5.5}$$

where $\varepsilon_1, \varepsilon_2$ are non-negative parameters.

The potential partial ordering $i \rightarrow j$ is termed an *admissible* one if there is not evidence that $i \rightarrow j$ cannot be selected as the locally most favorable replacement for the partial ordering $i \rightarrow x(i)$.

Let $\delta_{a,b}$ denote the *gain* obtained in $t(b)$ by introducing $a \rightarrow b$, such that

$$\delta_{a,b} = t(b) - \bar{t}(b) \tag{5.6}$$

where the new *ap* of node b can be expressed

$$\bar{t}(b) = \begin{cases} t(a) + T_{a,b} & \text{for } y = 0 \\ max\{r_b - 1, t(a) + c_{a,b}\} + p_b & \text{for } y = 1 \end{cases} \tag{5.7}$$

Formally, $i \rightarrow j$ for $x(i) \neq j$ is an admissible partial ordering whenever condition (5.8) is satisfied

$$\delta_{i,j} \geq \varepsilon_3 \tag{5.8}$$

where ε_3 is a given parameter (usually, a small negative value).

Note that $i \rightarrow j$ *cuts* the incumbent $\mathscr{H}$ so that the cycle $i \rightarrow sp(j \rightarrow i)$ is introduced. Node k breaks this cycle and, then, creates a new $\mathscr{H}$ that additionally improves the objective function. Formally, let

$$\Delta_{k'} = \begin{cases} T_{i,x(i)} - T_{i,j} + T_{y(j),j} - T_{y(j),k'} + T_{y(k'),k'} - T_{y(k'),x(i)} & \text{for } y = 0 \\ t(y(0)) - \bar{t}(y(0)) & \text{for } y = 1 \end{cases} \tag{5.9}$$

subject to

$$k' \in sp(x(j) \rightarrow 0) \tag{5.10}$$

where $\bar{t}(y(0))$ is given by (5.14) for $y = 0$ and (5.17)-(5.22) for $y = 1$. Choose k, such that

$$k = arg.max \ \{\Delta_{k'} > 0\} \tag{5.11}$$

and the new $\mathscr{H}$ satisfies the DFCs (5.3) and the PFCs (5.4). (Note that, by construction, $e_i \leq \bar{l}$, is always satisfied.)

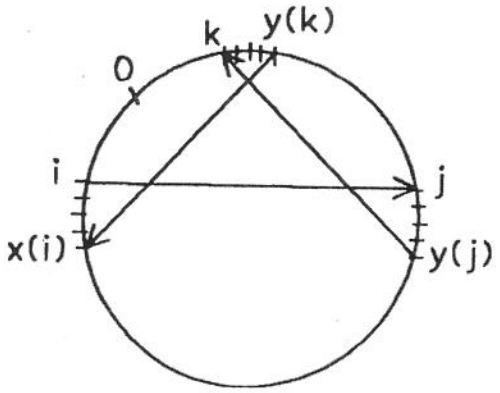

Figure 4. A TRIA change

Remark 1. The TRIA local search *does not* allow reverse partial ordering, so $j \prec k$.

Remark 2. Our TRIA implementation only allows $i \prec j$. Note that any TRIA-based Hamiltonian $\overline{\mathscr{H}}$ whose tuple $(\bar{i}, \bar{j}, \bar{k})$ is such that $\bar{k} \prec \bar{i}$ (see Figure 5), i.e.,

$$sp(0 \rightarrow y(\bar{j})) \rightarrow sp(\bar{k} \rightarrow \bar{i}) \rightarrow sp(\bar{j} \rightarrow y(\bar{k})) \rightarrow sp(x(\bar{i}) \rightarrow y(0)) \tag{5.12}$$

is the same Hamiltonian $\mathscr{H}$ (5.2) where $i = y(\bar{j})$, $j = \bar{k}$ and $k = x(\bar{i})$. So the tuple (i,j,k) to use is such that $i \prec j \prec k$.

Remark 3. Even for $\mathscr{H} = a \rightarrow x(a) \rightarrow b$, a TRIA change is allowed; namely, $a \rightarrow b \rightarrow x(a)$. Similarly, the EDD-based infeasible solution for the case shown in Tables 3 and 4 can be modified with a TRIA change, where $i = 1$, $x(i) = y(j) = 2$, $j = y(k) = 4$, $k = x(j) = 3$, such that the feasible $\mathscr{H} = 1 \rightarrow 4 \rightarrow 2 \rightarrow 3 \rightarrow 6 \rightarrow 5$ is found.

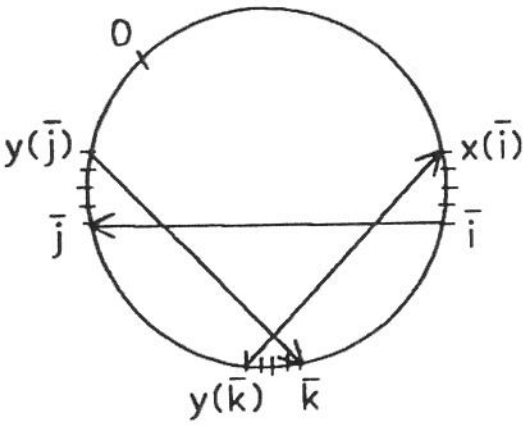

Figure 5. A TRIA change

The makespan of the new $\mathscr{H}$ will be

$$\bar{t}(y(0)) = t(y(0)) - \Delta \tag{5.13}$$

where $\Delta \equiv \Delta_1$.

Then, $\bar{t}(v)\ \forall v \in V$ can be expressed by (5.14) for $y = 0$ (no release dates) and (5.17)-(5.20) for $y = 1$.

$$\bar{t}(v) = t(v) - \phi_v \qquad \forall v \in V \tag{5.14}$$

where

$$\phi_v = \begin{cases} \delta_{ij} & v \in sp(j \to y(k)) \\ \delta_{ij} + \delta_{y(k),x(i)} & v \in sp(x(i) \to y(j)) \\ \Delta & v \in sp(k \to y(0)) \end{cases} \tag{5.15}$$

Note that the DFC (5.3) is not required to be analyzed for any node $v \in V$ such that condition (5.16) is satisfied.

$$\phi_v \geq max\ \{t(\ell) - d_\ell \quad \ell \in V\} \tag{5.16}$$

$$\bar{t}(j) = max\ \{r_j - 1, t(i) + c_{ij}\} + p_j \tag{5.17}$$

$$\bar{t}(v) = max\ \{r_v - 1, \bar{t}(y(v)) + c_{y(v),v}\} + p_v \tag{5.18}$$

$$\text{for}\quad v \in sp(x(j) \to y(k)) \cup sp(x(x(i)) \to y(j)) \cup sp(x(k) \to y(0))$$

$$\tilde{t}(x(i)) = max\ \{r_{x(i)} - 1,\ t(y(k)) + c_{y(k),x(i)}\} + p_{x(i)} \tag{5.19}$$

$$\tilde{t}(k) = max\ \{r_k - 1,\ t(y(j)) + c_{y(j),k}\} + p_k \tag{5.20}$$

The current $\mathcal{H}$ *does not* satisfy the PFCs (5.4) if the following condition is satisfied.

$$\exists a \in sp(x(i) \rightarrow y(j)) \quad \text{and} \quad \exists b \in sp(j \rightarrow y(k)) \quad \text{such that } (a,b) \in P \tag{5.21}$$

The order on which the *candidate* node k and the node b are chosen from $sp(j \rightarrow 0)$ to analyze condition (5.21) has a great influence in the computational effort required by the procedure outlined below. The effort is reduced when the order is $x(j),x(x(j)),...$ for node k and $j,x(j),...$ for node b. An appropriate order is necessary but not sufficient for avoiding unnecessary operations in the feasibility testing of the new $\mathcal{H}$. Given the above order for selecting node k, note that it is enough to restrict b to node $y(k)$ while analyzing condition (5.21). So, let $K_{i,j}$ denote the last *admissible* node for k. It can be expressed

$K_{i,j}$: Last node from $sp(x(j) \rightarrow 0)$ such that $\nexists a \in sp(x(i) \rightarrow y(j)) \mid (a,y(k)) \in P$ (5.22)

5.2. ROUGH ALGORITHM

Formally, the TRIA procedure is as follows.

Set $\tilde{t} := t(y(0))$

For $i \in sp(0 \rightarrow y(y(y(0))))$ such that condition (5.5) is satisfied

For $j \in sp(x(x(i)) \rightarrow y(0))$ such that condition (5.8) is satisfied, where $\delta_{i,j}$ as given by (5.6)

For $k \in sp(x(j) \rightarrow K_{i,j})$ (Analyze the Hamiltonian (5.2):)

For $\gamma = 0$ (no release dates) :

Compute Δ (5.9)

If $\Delta > 0$

and ($\delta_{i,j}$ *does* satisfy condition (5.16) *or* $\nexists v \in sp(j \rightarrow y(k))$ such that DFC (5.3) *is not* satisfied)

and ($\delta_{i,j} + \delta_{y(k),x(i)}$ *does* satisfy condition (5.16) *or* $\nexists v \in sp(x(i) \rightarrow y(j))$ such that DFC (5.3) *is not* satisfied)

then update $\tilde{t} := \bar{t}(y(0))$ and save nodes' tuple (i,j,k)

(Note. DFC (5.3) is analyzed for $\bar{t}(v)$ as given by (5.14).)

For $\gamma = 1$ (release dates) :

If $\nexists v \in \mathscr{H}$ (5.2) such that DFC (5.3) *is not* satisfied, where $\bar{t}(v)$ as given by (5.17)-(5.20)

and $\bar{t}(y(0)) < \tilde{t}$

then update $\tilde{t} := \bar{t}(y(0))$ and save nodes' tuple (i,j,k)

Endfor k

Endfor j

Endfor i

If $\tilde{t} \neq t(y(0))$ then update current sequence with incumbent (5.2) and begin again.

6. A QUAD PROCEDURE FOR IMPROVING FEASIBLE SOLUTIONS

6.1. FRAMEWORK

This section describes a QUAD-based procedure for improving the initial feasible $\mathscr{H}$ provided that $n \geq 4$. (The notation used by QUAD is consistent with the notation introduced above.) QUAD works on the nodes tuple (i,j,k,g), such that the new partial orderings (see Figure 6) are $\bar{x}(i) := j$, $\bar{x}(y(j)) := x(i)$, $\bar{x}(k) := g$ and $\bar{x}(y(g)) := x(k)$. Then, the QUAD search seeks to improve the makespan $t(y(0))$ of the current $\mathscr{H}$ (5.1) by finding the nodes' tuple (i,j,k,g) such that the new sequence

$$sp(0 \rightarrow i) \rightarrow sp(j \rightarrow k) \rightarrow sp(g \rightarrow y(j)) \rightarrow sp(x(i) \rightarrow y(g)) \rightarrow sp(x(k) \rightarrow y(0)) \qquad (6.1)$$

gives a better makespan, i.e., $\bar{t}(y(0)) < t(y(0))$ and satisfies the DFCs (5.3) and PFCs (5.4), where the expression for $\bar{t}(v)$ is given below.

As above, a node, say i is termed an *admissible* node if there is not evidence that it cannot be selected as the first one in the nodes' tuple. Formally, node i is an admissible node whenever condition (5.5) is satisfied.

Note that usually no TRIA, QUAD change reverses any partial ordering (i.e., $i \rightarrow j$ is not replaced by $j \rightarrow i$); note also that $T_{a,b} \neq T_{b,a}$ and the reverse change could be computational expensive. In any case, we recommend to use recursively the sequence TRIA-QUAD until no more improvement is possible.

Let $\Delta_{a,b}$ denote the *gain* obtained by introducing $a \rightarrow b$, such that

$$\Delta_{a,b} = \begin{cases} T_{a,x(a)} - T_{a,b} + T_{y(b),b} - T_{y(b),x(a)} & \text{for } \gamma = 0 \\ \delta_{a,b} + \delta_{y(b),x(a)} & \text{for } \gamma = 1 \end{cases} \tag{6.2}$$

where

$$\delta_{a,b} = t(b) - [max\ \{r_b - 1, t_a + c_{a,b}\} + p_b] \tag{6.3}$$

and

$$\delta_{y(b),x(a)} = t(x(a)) - [max\ \{r_{x(a)} - 1, t(y(b)) + c_{y(b),x(a)}\} + p_{x(i)}] \tag{6.4}$$

The potential partial ordering $i \rightarrow j$ is termed admissible whenever condition (6.5) is satisfied

$$x(i) \neq j \text{ and } x(x(i)) \neq j \text{ and } \Delta_{i,j} > \varepsilon_4 \tag{6.5}$$

where ε_4 is a given parameter (usually, a small negative value).

It is assumed that the QUAD change based on $i \rightarrow j | \Delta_{i,j} \leq \varepsilon_4$ is not likely to produce a better solution. On the other hand, we may have some expectation of improving the solution for $\Delta_{i,j} > \varepsilon_4$. Note that k and g are the nodes whose partial orderings $k \rightarrow g$ and $y(g) \rightarrow x(k)$ produce a new $\mathscr{H}$ by *cutting* the cycles produced by $i \rightarrow j$ (see Figure 6). Formally, k and g must satisfy the following conditions

$$k \in sp(j \rightarrow y(0)) \tag{6.6}$$

$$g \in sp(x(x(i)) \rightarrow y(j)) \tag{6.7}$$

$$\Delta \equiv \begin{cases} \Delta_{ij} + \Delta_{kg} > 0 \quad \text{for} \quad \gamma = 0 \\ t(y(0)) - \bar{t}(y(0)) > 0 \quad \text{for} \quad \gamma = 1 \end{cases} \tag{6.8}$$

where the new makespan $\bar{t}(y(0))$ is given by (6.13), and the new $\mathscr{H}$ (6.1) should satisfy the DFCs (5.3) and PFCs (5.4).

Remark 1. Even for $\mathscr{H} = a \to x(a) \to y(b) \to b$, a QUAD change is allowed; namely, $a \to b \to y(b) \to x(a)$. We do not allow any other reverse change.

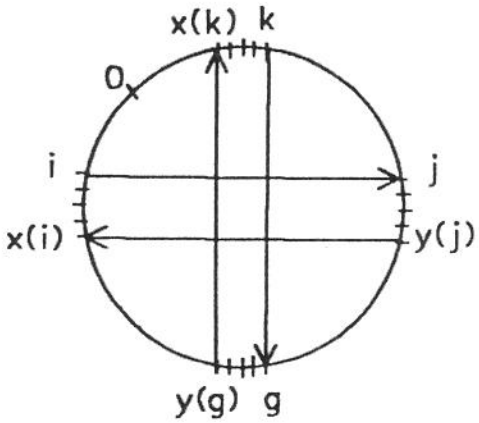

Figure 6. A QUAD change

The new *ap* of node v, $\bar{t}(v)\ \forall v \in V$ can be expressed by (6.9) for $\gamma = 0$ (no release dates) and (6.12)-(6.16) for $\gamma = 1$.

$$\bar{t}(v) = t(v) - \phi_v \qquad v \in V \tag{6.9}$$

where

$$\phi_v = \begin{cases} \delta_{ij} & v \in sp(j \to k) \\ \delta_{ij} + \delta_{kg} & v \in sp(g \to y(j)) \\ \delta_{ij} + \delta_{kg} + \delta_{y(j),x(i)} & v \in sp(x(i) \to y(g)) \\ \Delta & v \in sp(x(k) \to y(0)) \end{cases} \tag{6.10}$$

Note that the DFC (5.3) is not required to be analyzed for any node $v \in V$ such that condition (6.11) is satisfied.

$$\phi_v \geq max\ \{t(\ell) - d_\ell \quad \ell \in V\} \tag{6.11}$$

$$\bar{t}(j) = max\ \{r_j - 1, t(i) + c_{i,j}\} + p_j \tag{6.12}$$

$$\bar{t}(v) = max\ \{r_v - 1, \bar{t}(y(v)) + c_{y(v),v}\} + p_v \tag{6.13}$$

$$\text{for } v \in sp(x(j) \rightarrow k) \cup sp((x(g)) \rightarrow y(j)) \cup sp(x(x(i)) \rightarrow y(g)) \cup sp(x(x(k)) \rightarrow y(0))$$

$$\bar{t}(g) = max\ \{r_g - 1, t(k) + c_{k,g}\} + p_g \tag{6.14}$$

$$\bar{t}(x(i)) = max\ \{r_{x(i)} - 1, t(y(j)) + c_{y(j),x(i)}\} + p_{x(i)} \tag{6.15}$$

$$\bar{t}(x(k)) = max\ \{r_{x(k)} - 1, t(y(g)) + c_{y(g),x(k)}\} + p_{x(k)} \tag{6.16}$$

The current $\mathcal{H}$ *does not* satisfy the PFCs (5.4) if any of the following conditions is satisfied

$$\exists a \in sp(x(i) \rightarrow y(j)) \text{ and } \exists b \in sp(j \rightarrow k) \ni (a,b) \in P \tag{6.17}$$

$$\exists a \in sp(x(i) \rightarrow y(g)) \text{ and } \exists b \in sp(g \rightarrow y(j)) \ni (a,b) \in P \tag{6.18}$$

Note that $sp(j \rightarrow k) \prec sp(g \rightarrow y(j)) \prec sp(x(i) \rightarrow y(g))$ for the new $\mathcal{H}$ (6.1). An efficient implementation of the QUAD approach should include a mechanism very similar to the mechanism outlined at the end of Section 5. The order for analyzing the candidate node k and the node b in (6.17) is $j,x(j),...$ Note that, in this case, it is enough to restrict b to node k while analyzing condition (6.17). The order for analyzing the candidate node g and the node b in (6.18), *independently* of the node k, is $y(j),y(y(j)),...$ So, let $B_{i,j}$ and $K_{i,j}$ denote the last *admissible* nodes for nodes g and k, respectively. They can be expressed

$B_{i,j}$: Last node from $pp(y(j) \rightarrow x(x(i))) \ni \nexists a \in sp(x(i) \rightarrow y(K_j)) \mid (a, K_j) \in P$ (6.19)

$K_{i,j}$: Last node from $sp(j \rightarrow y(0)) \ni \nexists a \in sp(x(i) \rightarrow y(j)) \mid (a,k) \in P$ (6.20)

6.2. ROUGH PROCEDURE

Formally, the QUAD procedure is as follows.

Set $\tilde{t} := t(y(0))$

For $i \in sp(0 \rightarrow y(y(y(y(0)))))$ such that condition (5.5) is satisfied

For $j \in sp(x(x(x(i))) \rightarrow y(0))$ such that condition (6.5) is satisfied, where $\delta_{i,j}$ as given by (6.2)

Compute $B_{i,j}$ (6.19) and $K_{i,j}$ (6.20).

For $k \in sp(j \rightarrow K_{i,j})$ such that $\nexists a \in sp(x(i) \rightarrow y(j)) \mid (a,k) \in P$

For $g \in pp(y(j) \rightarrow B_{i,j})$ (Analyze the Hamiltonian (6.1):)

For $y = 0$ (no release dates) :

Compute Δ (6.8)

If $\Delta > 0$

and ($\delta_{i,j}$ *does* satisfy condition (6.11) *or* $\nexists v \in sp(j \rightarrow k)$ such that DFC (5.3) *is not* satisfied)

and ($\delta_{i,j} + \delta_{k,g}$ *does* satisfy condition (6.11) *or* $\nexists v \in sp(g \rightarrow y(j))$ such that DFC (5.3) *is not* satisfied)

and ($\delta_{i,j} + \delta_{k,g} + \delta_{y(0),x(0)}$ *does* satisfy condition (6.11) *or* $\nexists v \in sp(x(i) \rightarrow y(g))$ such that DFC (5.3) *is not* satisfied)

then update $\tilde{t} := \bar{t}(y(0))$ and save nodes' tuple (i,j,k,g)

(Note. DFC (5.3) is analyzed for $\bar{t}(v)$ as given by (6.9).)

For $y = 1$ (release dates) :

If $\nexists v \in \mathcal{N}$ such that DFC (5.3) *is not* satisfied, where $\bar{t}(v)$ as given by (6.12)-(6.16)

and $\bar{t}(y(0)) < \tilde{t}$

then update $\tilde{t} := \bar{t}(y(0))$ and save nodes' tuple (i,j,k,g)

Endfor g

Endfor k

Endfor j

Endfor i

If $\tilde{t} \neq t(\tilde{s}(0))$ then update current sequence with incumbent (6.1) and begin again.

7. AN ILLUSTRATIVE EXAMPLE

As an illustrative example for the proposed algorithm, let us consider the case drawn from a production line in IBM Vimercate plant. The problem consists of sequencing the computer cards to be processed on an assembly machine. The cards belong to 5 part types. Table 5 gives the production volume (i.e., number of cards), processing time and setup time per part type.

The processing and setup times are given in Time Machine Units (TMU), where 10000 TMU = 1 hour (approx.). The production plan is based on 10 shifts with 6 hours each. The total number of cards to be processed along the planning horizon is 139. Any card from part type B must be processed before any card from part type C; so, there are 476 precedence relationships. In this case, $|P| = 3.4\,n$ (approx.), where $n = |V| = 139$ nodes (i.e., cards) and $|A| = 14049$ arcs.

Table 5. Production data

Part numbers	# of cards	time	
		process	setup
A	41	3270	4200
B	28	4644	0
C	17	3834	0
D	10	3336	4200
E	43	2518	4200

The setup time $c_{i,j}$ is obtained as follows. $c_{i,j} = 0$ for $f(i) = f(j)$ and, otherwise, $c_{i,j} = s_{f(j)}$ for all $i \in V$, where $f(v)$ gives the part type of card v and s_ℓ gives the setup time for part type ℓ =A,B,C,D,E.

Table 6 gives the cards that are released at the beginning of each hour of the planning horizon. Table 7 gives the cards that are due at the end of each hour.

The algorithm described in this paper has been written in C language, and run on an IBM PS/2 model 70-386 with Microsoft C compiler 5.1 under OS/2. See in [5] the implementational details.

The results are as follows: Preprocessing: 27% and 32% reduction on the cardinality of the arc sets A and P, respectively, and an 8% reduction on the due'release dates gap. The initial solution does not satisfy the due dates for 12 nodes, and the completion time is 644451 TMU. The local search *restricted* TRIA-1 requires 12 major iterations and obtains a feasible solution whose completion time is 599405 TMU. The local searches TRIA and QUAD improve the feasible solution. TRIA requires 20 major iterations (e.g., 20 interchanges from (5.1) to (5.2) have been performed) and QUAD requires 6 major iterations. The completion time of the best solution is 538143 TMU. One more TRIA major iteration gives our currently best

solution, 537219 TMU. A 10.37% improvement has been obtained. The total CPU time is 1842 secs. See Table 8.

Table 6. Release dates for case given in Table 5

Hour(shift)	Card (part type)
1(1)	1-5(A),87-88(D),97-100(E)
7(2)	6-9(A),42-50(B),70-75(C),89-90(D),101-104(E)
13(3)	10-15(A),51-60(B),76-77(C),105-110(E)
19(4)	16-23(A),61-66(B),78-84(C),111-115(E)
25(5)	24-26(A),67-69(B),85-86(C),94-94(D),116-124(E)
37(7)	27-36(A),95-96(D),125-128(E)
43(8)	37-41(A),129-132(E)
49(9)	133-139(E)

Table 7. Due dates for case given in Table 5

Hour(shift)	Card (part type)
6(1)	1-5(A),87-88(D),97-100(E)
12(2)	6-9(A),42-48(B)
18(3)	10-15(A),49-50(B),89-90(D)
30(5)	16-26(A),51-55(B),101-110(E)
36(6)	56-60(B)
42(7)	61-69(B),70-77(C),111-115(E)
48(8)	27-31(A),91-96(D)
54(9)	78-84(C),116-132(E)
60(10)	32-41(A),85-86(C),133-139(E)

Table 8. Results for case given in Table 5-7

Solution type	# of major iterations	Completion time		CPU time (secs)
		TMU	Improve (%)	
EDD	-	644451	-	-
Restricted TRIA-1	12	599405	6.99	9
TRIA	20	545759	8.94	1069
QUAD	6	538143	1.40	734
TRIA again	1	537219	0.17	30

Elsewhere [6] we report our tightest lower bound on the optimal solution. Here we use an apparently weak bound that is computed by summing up the total proc-

essing time of the 139 cards and the minimum required setup time. We assume that all cards that belong to the same part type will be sequenced in a row and, then, only one setup per part type is needed. See Table 3. The value of this lower bound is 493514 TMU. Then, the optimality gap is 8.85%

CONCLUSIONS

In this work we have presented a 0-1 model for the Hamiltonian path problem with release and due dates and precedence relationships. We have also outlined the basic steps of a procedure for obtaining feasible solutions, and the main ideas of two local search procedures for improving the initial solution. In a companion paper we describe a cut generation LP algorithm for obtaining lower bounds on the optimal solution and, in fact, assessing the statistical performance of the approximate approach presented here. An extensive computational study is in progress. Based on available results, we believe that methodologies as the one described here are practical ways to deal with large-scale real-life cases.

REFERENCES

[1] N. Ascheuer, L.F. Escudero, M. Grötschel and M. Stoer, On identifying in polynomial time violated subtour elimination and precedence forcing constraints for the sequential ordering problem, **SIAM J. on Optimization** (submitted for publication).

[2] L.F. Escudero, An inexact algorithm for the sequential ordering problem, **European J. of Operational Research** 37 (1988) 236-253.

[3] L.F. Escudero, On the implementation of an algorithm for improving a solution to the sequential ordering problem, **Trabajos de Investigacion-Operativa** 3 (1988) 117-140.

[4] L.F. Escudero and A. Sciomachen, Framework of an LP cuts-based algorithm for the sequential ordering problem with time windows and precedence relationships,

RC-15597, IBM Research, T.J. Watson Research Center, Yorktown Heights, NY, 1990.

[5] L.F. Escudero and A. Sciomachen, An implementation of an approximate algorithm for finding a feasible solution for the sequential ordering problem, Research Report xxxx, Faber, Milano, 1991.

[6] L.F. Escudero, M. Guignard, K. Malik and A. Sciomachen, On Lagrangian-based lower bounds for the sequential ordering problem with time windows and precedence relationships, in preparation.

[7] P.C. Kanellakis and Ch.C. Papadimitrou, Local search for the asymmetric travelling salesman problem, **Operations Research** 28 (1980) 1086-1099.

[8] S. Martello, An enumerative algorithm for finding Hamiltonian circuits in a directed graph, **ACM Transactions on Mathematical Software** 9 (1983) 131-138.