DÉPARTEMENT DE GÉNIE INDUSTRIEL

Rapport Technique EP76-R-14
Classification: Library of Congress No.....

The Time-Dependent Traveling Salesman Problem and Application to the Tardiness Problem in One-Machine Scheduling.

par

Jean-Claude Picard
et
Maurice Queyranne

Juin 1976

Ecole Polytechnique de Montréal

C.P. 501
Snowdon
Montréal 248

he Time-Dependent Traveling Salesman and Application to the Tardiness Problem in One-Machine Scheduling.

par

Jean-Claude Picard

et

Maurice Queyranne

Département de Génie Industriel

Ecole Polytechnique

* This research was supported by The National Research Council of Canada, Grants CNRC A8528 and RD804.

THE TIME-DEPENDENT TRAVELING SALESMAN PROBLEM AND APPLICATION TO THE TARDINESS PROBLEM IN ONE-MACHINE SCHEDULING.

Jean-Claude PICARD

Maurice QUEYRANNE

Department of Industrial Engineering

Ecole Polytechnique, Montréal (Canada)

ABSTRACT:

The "Time-Dependent Traveling Salesman Problem", defined by K. Fox is a scheduling problem in which n jobs have to be processed at minimum cost on a single machine. The "set-up costs", associated with each job transition, depend not only on the two successive jobs, but also on their position ("time") in the sequence. The approach described here utilizes a shortest path problem on a related multipartite network, combined with subgradient optimization and and some implicit enumeration.

Minimizing the tardiness costs in one-machine scheduling (in which the cost is a non-decreasing function of the completion time of each job) is then approached by this method. A "Time-Dependent Traveling Salesman Problem" is used to define a new branch and bound algorithm. Computational results are given for the weighted tardiness problem and are strikingly better than previous known methods.

INTRODUCTION

This paper describes a general model for several optimum permutation and related problems. This model, called the Time-Dependent Traveling Salesman Problem (TDTSP) is a generalization of the Traveling Salesman (TSP) and Assignment problems. In this problem, the costs of transition depend not only on the two successive locations in a tour, but also on the period when these transitions occur.

In PART ONE, a framework for an algorithm is given, using shortest path iterations and subgradient optimization within a branch and bound structure. A general and powerful dominance test is introduced to alleviate the enumeration effort.

An application of the TDTSP to a general One-Machine Sequencing problem is given in PART TWO. This problem consists of minimizing the total tardiness costs in One-Machine Scheduling where the cost is a non-decreasing function of the completion time of each job.

A multipartite network is defined in order to apply the TDTSP general algorithm. Efficient reductions in this network are described. A branch and bound algorithm is then derived and computational results give indication of the efficiency of this approach for the Weighted Tardiness problem

Finally, further applications of the TDTSP to several classes of problems are outlined in the conclusion.

PART ONE Set-up Costs: the Time-Dependent Traveling Salesman Problem

1.1 Definitions

The following one-machine sequencing problem is called the Time-Dependent Traveling Salesman Problem (TDTSP).

Consider a set $\mathcal{J}$ of n jobs, denoted by $J_1,\ldots,J_n$, to be performed on a single machine and set-up costs C_{ij}^t occuring when the job J_i, processed in the t^{th} position, is followed by the job J_j (processed in the $(t+1)^{st}$ position). The machine is in a given "initial state", denoted by 0, before the job processing. It has to be returned to a given "final state", denoted by $(n+1)$, after the job processing, and initial and final set-up costs $C_{o,i}^o$ and $C_{i,n+1}^n$ are also given. The problem is to find a sequence $J_{w(1)},\ldots,J_{w(n)}$ which minimizes the total set-up cost $C_{(w)}$, defined by

$$C_{(w)} = C_{o,w(1)}^o + C_{w(1),w(2)}^1 + \ldots + C_{w(n-1),w(n)}^{n-1} + C_{w(n),n+1}^n \quad (1.1)$$

Example 1: Suppose n = 6 and the set-up costs given in Table I (The crosses correspond to infinite costs i.e. to forbidden transitions).

The cost of the sequence w = (0,1,2,3,4,5,6,7) is

$$C_{(w)} = C_{o,1}^o + C_{1,2}^1 + C_{2,3}^2 + C_{3,4}^3 + C_{4,5}^4 + C_{5,6}^5 + C_{6,7}^6$$

$$= 0 + 0 + 0 + 98 + 180 + 264 + 0$$

$$= 542$$

Phase 1

	1	2	4
0	0	0	0

Phase 2

	1	2	4	5	6
1	x	0	0	x	x
2	0	x	0	x	x
4	0	0	x	0	0

Phase 3

	1	2	3	4	5	6
1	x	72	0	0	0	0
2	0	x	0	0	0	0
4	0	72	x	x	0	0
5	84	198	x	x	x	0
6	119	243	x	x	100	x

Phase 4

	1	2	3	4	5	6
1	x	306	20	x	40	0
2	175	x	20	x	40	0
3	x	x	x	98	x	x
4	x	x	20	x	40	0
5	175	306	x	x	x	0
6	210	351	x	x	220	x

Phase 5

	1	2	3	5	6
1	x	585	150	350	110
2	392	x	150	350	110
3	x	x	x	180	0
4	x	x	x	180	0
5	392	585	150	x	110
6	392	585	175	350	x

Phase 6

	3	5	6
1	305	x	x
2	305	x	x
3	x	490	264
5	305	x	264
6	305	490	x

Phase 7

	7
3	0
5	0
6	0

Table I

Remarks:

1 - The entries $C^t_{i,i}$ need not to be defined.

2 - Problems with unspecified initial (resp.final) state are formulated in the same way, using $C^o_{oi} = 0$ (resp. $C^n_{i,n+1} = 0$) for all i.

3 - When set-up costs are not time-dependent, that is

$$C^t_{ij} = C_{ij} \qquad \text{all } t$$

for all (i,j),the problem is reduced to the classical traveling salesman problem (TSP)

4 - When set-up costs are not dependent on the destination job, that is

$$C^t_{ij} = C^t_i \qquad \text{all } j$$

for all (i,t), the problem is reduced to the classical assignment problem (AP).

The TDTSP was defined by Kenneth R. Fox in his dissertation [7] entitled "Production Scheduling on Parallel Lines with Dependencies", and it was illustrated with examples from the brewing industry. The following is a variation on a classical illustration ([4 , p.53]) of the TSP.

Example 2: Consider a paint factory, producing on a weekly basis five different colors of paint, one per day. The machine has to be cleaned at each color change. The changeover is accomplished by a night staff which works primarily on other fixed tasks in the factory, within a given schedule. This leaves a fixed amount of time each night for the changeover, but this time may vary from day to day. Because of these

varying available changeover times, production factors such as manpower and chemical products may be used at different levels. So the set-up costs are dependent, not only on the color being removed and the color for which the machine is being prepared, but also on which night the operation is performed.

Fox's approach is an integer-programming formulation with about n^3 binary variables and 4n constraints, which proved inefficient for problems with $n > 10$.

1.2 Shortest path approach

It seems quite natural to define a "multipartite graph" G = (V,A) (cf[19]) associated to this problem: the node set V consists of nodes

(0),
(i,t) for i,t = 1,2,...,n
(n+1)

where the node (i,t) represents the possible execution of job J_i in phase t.

The arcs in A are:

initial arcs (0,(i,1)) with length $C^o_{o,i}$

transition arcs ((i,t),(j,t+1)) with length $C^t_{i,j}$ $(i \neq j)$

final arcs ((i,n),n+1).

The multipartite graph associated with the 6-jobs TDTSP of example 1 is pictured in Fig. 1.

A path P in G joins 0 to (n+1) through n transition nodes, one in each phase.

If every job in $\mathcal{J}$ appears exactly once among the n transition nodes in a path P, then P represents a feasible sequence w, and its length ℓ^P is the cost C (w) of this sequence.

Such a path will be called a sequence path. If the shortest path in the network is a sequence path, then the corresponding sequence is an opti-

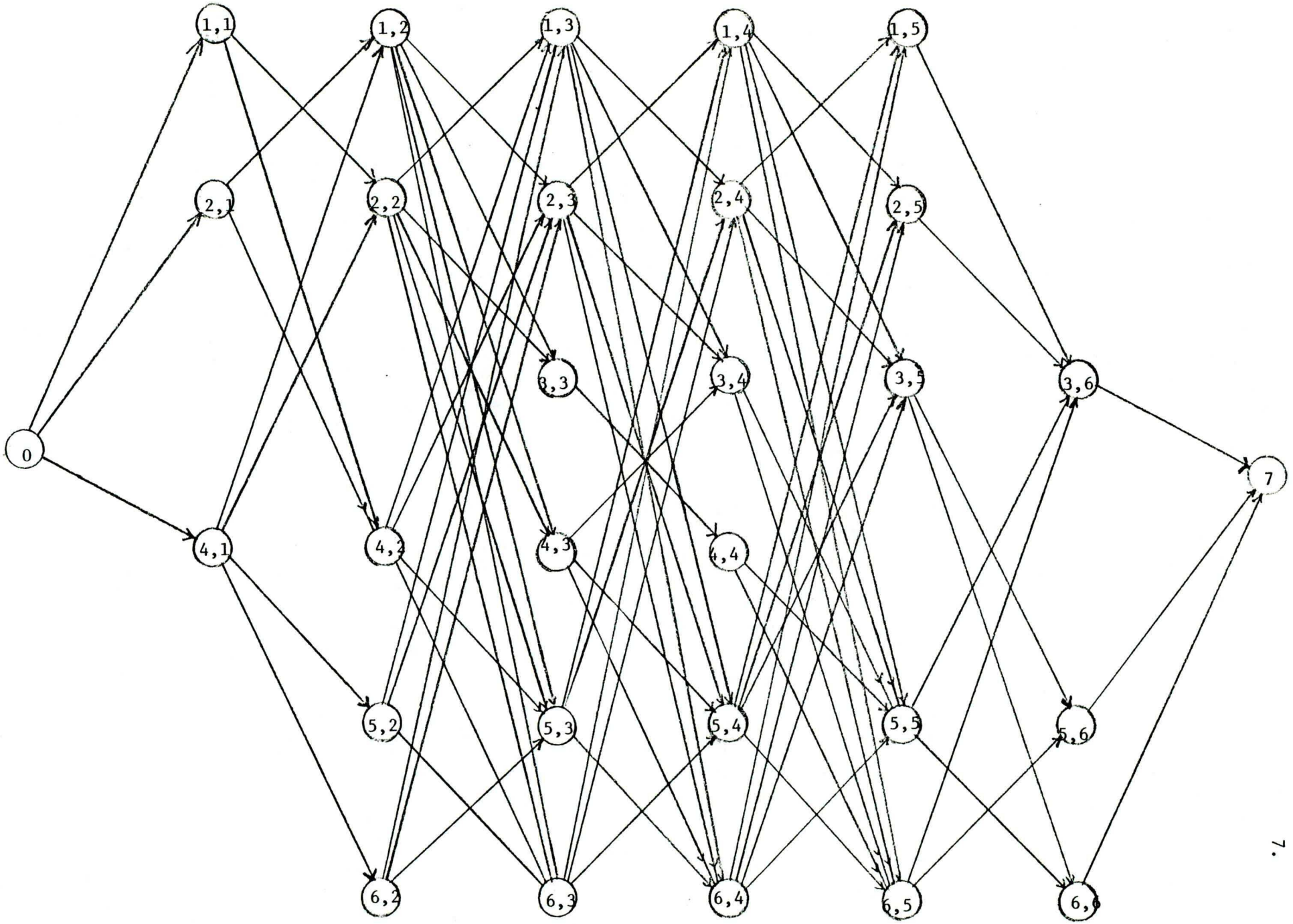

1,1
1,2
1,3
1,4
1,5
2,1
2,2
2,3
2,4
2,5
3,3
3,4
3,5
3,6
0
7
4,1
4,2
4,3
4,4
5,2
5,3
5,4
5,5
5,6
6,2
6,3
6,4
6,5
6,6

FIGURE 1

mal solution to TDTSP; otherwise, it is advisable to perturb the problem in order to force the shortest path to visit the forsaken jobs and to avoid the jobs which were too much visited.

A possible solution is the use of penalties $\P_j$ incurred by a path each time a node representing job J_j is visited. These penalties may be inserted in the network by adding $\P_i$ to the length of all arcs coming from all nodes (i,t) associated with J_i:

$$\overline{C}^t_{ij} = C^t_{ij} + \P_i \qquad (1\text{-}2)$$

(it is also possible to add $\P_j$ to all edges going into (j,t) or to use a convex combination of these two schemes, for example by redefining

$$\overline{C}^t_{ij} = C_{ij} + \tfrac{1}{2}\P_i + \tfrac{1}{2}\P_j$$

Note first that a same constant may be substracted to all penalties, so it is possible to consider that the following relation always holds:

$$\sum_{j=1} \P_j = 0 \qquad (1\text{-}3)$$

Denoting by a^P_i the number of occurrences of nodes (i,t) (for $t = 1,2,\ldots,n$) representing job J_i in the path P, the length of P becomes, in the perturbed network:

$$\overline{\ell}\,(P) = \ell(P) + \sum_i a^P_i \P_i \qquad (1\text{-}4)$$

If P is a sequence path, associated with w, then

$$\overline{\ell}\,(P) = \ell(P) = C(w) \qquad (1\text{-}5)$$

Denoting by C* the cost of the optimal sequence, and by P_π the shortest path in the network perturbed by π, we have:

$$\overline{\ell}\,(P_\pi) \leqslant C^* \qquad (1\text{-}6)$$

The "best" penalties π are solutions of the following maximization problem:

$$\max_{\pi} \{W(\pi) \;/\; \sum_i \pi_i = 0\}$$

where $$W(\pi) = \overline{\ell}\,(P_\pi) = \min_P \left(\ell(P) + \sum_i a_i^P \pi_i\right)$$

This problem may be stated as a linear programming problem (P):

$$(P) \quad \begin{cases} \text{Max } W \\ \text{s.t.} \quad W - \sum_i a_i^P \pi_i \leqslant \ell(P) \qquad \text{all } P \\ \sum_i \pi_i = 0 \\ W,\ \pi_i \ \text{unconstrained.} \end{cases}$$

Note that (P) has an enormous number $(1 + n(n-1)^{n-1})$ of constraints.

The dual (D) of (P) is:

$$(D) \quad \begin{cases} \text{Min} \sum_P \ell(P)\, x_P \\ \sum_P x_P = 1 \\ \sum_P a_i^P x_P + y = 0 \qquad i = 1,2,\ldots,n \\ X \geqslant 0 \end{cases}$$

Note that if $\sum_i \pi_i^k = 0$ then $\sum_i \pi_i^{k+1} = 0$ follows from $\sum_i a_i^p = n$; so (1-3) holds whenever it holds for the initial π^o.

The subgradient iterations are performed until one of the following conditions holds:

(i) $a^{P_k} = \underline{1}$ so P_k is the optimal solution to TDTSP.

or

(ii) p_2 iterations have been performed (p_2 is a second fixed parameter).

When the outcome of the iteration scheme is in (ii), $\overline{W}_k$ is a lower bound for the optimal value of the objective function of (P).

This bound will be used to define a branch and bound structure for solving (P).

1.4 Branch and bound structure.

The set of all the feasible sequences in the current problem (P) is partitioned with respect to the last job to be processed. If $k_1, k_2 \ldots, k_s$ are the indices of the jobs for which the arcs $((k_r,n),(n+1))$ exist, then s subproblems $P_1, P_2, \ldots, P_s$ are defined by fixing the job J_{k_r} as the last job in P_r, for $r=1,2,\ldots,S$. The problem (P) is called the "father problem" and the subproblems P_r are called its "sons". The father being a TDTSP with size n, the sons are also TDTSP with size n-1, and some information (bounds, penalties,...) obtained for the father problem may be used to handle its sons.

The shortest path iterations in (P) may be performed by using a dynamic programming algorithm: if f(i,t) denotes the shortest distance from the origin (0) to node (i,t), then the following recurrence relation holds

$$f(i,t) = \text{Min } \{f(j,t-1) + C_{ji}^t\} + \pi_i$$

in which the minimum is taken over the set of indices j for which $f(j, t-1)$ and C^t_{ji} are defined, the initial value being $f(o,o) = f(o) = 0$. The length of the required shortest path is

$$f(n+1) = \text{Min } \{f(k_r,n) + C^n_{k_r,n+1} \}$$

As a by-product of the algorithm, these values $b_r = f(j_r,n) + C^n_{k_r,n} + 1$ provide lower bounds on the length of minimum sequence paths with J_{k_r} as the last job. While performing the iterations in (P), it is advisable to store in B_r the maximum of the b_r's computed for the successive penalties π. These values B_r are called the "implicit bounds" for P_r.

The implicit bounds allow early fathoming of the subproblems P_r if $B_r > UB$ (UB is the length of the best known sequence path). The exploration strategy called LIFO (or depth-first search, or backtracking) will make use of a subsequent ranking of the remaining P_r's, according to a non-increasing order of their implicit bounds. This exploration strategy may be preferred because of its predictible storage requirement, in contrast with the least bound strategy (jump-tracking).

Another implication of the use of the implicit bounds is called "modified subgradient". After some shortest path iterations, the values of B_r are large enough to make the shortest path in the whole network of no value in itself. It appears more advisable to use, as a direction of modification for the penalties, the shortest path ending in the node (k_r,n) with least bound B_r. In this manner, a more specific attempt is made to improve the worst (least) implicit bound, and this could be seen as taking advantage of a look-ahead power over the sons of the current problem. The experiments made during the designing phase of the algorithms for several applications (TSP, Tardiness problem), showed that this modification was especially worthwile, as much for earlier fathoming as for tightening the bounds.

1.5 Dominance Test.

In order to reduce the size of the enumeration, a dominance test is introduced for fast detection of unprofitable subproblems. This test is based on attempts to improve a current (partial) solution by inserting a segment of the sequence between two other successive jobs.

Suppose that a subproblem (P) is defined by fixing the (n-k) last jobs, say J_{k+1}, J_{k+2},...,J_n. Its sons will be defined by fixing the k^{th} job. Consider J_r as a possible last job. If the partial sequence $J_r, J_{k+1}, J_{k+2}, \ldots, J_n$ can be improved, without altering the k first phases, then the corresponding subproblem will not contain the optimal sequence.

Improvements may be checked by inserting J_{k+1} between J_{k+2} and J_{k+3}, then between J_{k+3} and J_{k+4},..., finally between J_n and the final state. As soon as any improvement occurs, the corresponding subproblem is disregarded.

Otherwise, tigher tests may be performed by inserting the segment J_{k+1}, J_{k+2} between J_{k+3} and J_{k+4}, and so on. The number of jobs in the segment may be bounded by a fixed parameter or by its largest value n-k+1 (corresponding to the insertion of the segment $J_{k+1}, \ldots, J_{n-1}$ after J_n). If the same test has been applied in the previous branching steps and if no improvement occurs, then the partial sequence J_r, J_{k+1}, ..., J_n is a locally optimal sequence with J_r fixed as "first"job, and the time periods being from k to n+1.

These tests could be performed before the shortest path iterations in order to reduce the number of possible last jobs and make the shortest path iterations and the related implicit bounds more accurate.

The value of this dominance test, which can be extended to a large number of enumeration algorithms, was widely demonstrated in the TSP and the Tardiness Cost problem applications of the model: it resulted in a considerable reduction in the size of the enumeration and in the computer time, and this to the extent that it allowed the solution of problems which were unsolvable (in a "reasonable" time) without it.

The idea of improving a given solution by insertion was used by Emmons[5] to introduce his precedence relation. A special case, where the segment is reduced to one job and the insertion is tried just on both sides of this job is called "adjacent pairwise interchange". See[2]where a good discussion of this kind of method is given. It appears that the idea originates from the work by Reiter and Sherman[17], describing their general heuristic method for discrete optimization. In particular, it provides a good heuristic method for obtaining a good initial solution. Its value is well shown for the specific applications to the TSP and the Weighted Tardiness problem.

The authors will not present computational results for the general TDTSP, since it is mainly brought up as a model for solving a general class of optimal permutation and related problems. Moreover, defining test data for such a general problem by random sampling of the entries C^t_{ij} has little meaning and it appears that specialized problems are much more difficult than random ones.

b) $C_i(t) = \alpha_i(t - d_i)$

The problem is to minimize the weighted lateness; it is solved by WSPT sequencing (weighted shortest processing time) i.e. by a non-decreasing order of p_i/α_i . See [2].

c) $C_i(t) = \alpha_i \ \text{Max}\ \{0;\ t - d_i\} + \beta_i t$

This more general problem is to minimize the sum of (weighted) tardiness costs and flow-time costs.

The MTCP received much attention [8], [18] and it is considered as a difficult combinatorial problem.

In sections 2-2 and 2-3 the cost functions are supposed to be any non-decreasing functions.

2.2 TDTSP Formulation.

Earlier results, originated with the work of Emmons [5], are worthwile for defining a "precedence relation", that is a partial order on the job set, such that an optimum sequence would be found among the extensions of this partial order. The reader is referred to sections 2-2 and 2-3 of [18] for the definition and computation of such a precedence relation.

This relation will be denoted by P and iPj means that job J_i must be performed before job J_j, according to P; in this case, job J_j is called a descendant of job J_i and job J_i is called an ascendant of J_j. A_i and B_i will respectively denote the set of indices of descendants and of ascendants of job J_i (in the following, A_i and B_i will denote indifferently set of indices and sets of the corresponding jobs).

Letting S(i,j,t) be the set of sequences which are extensions of the given precedence relation and in which the t^{th} job is J_i and the $(t+1)^{st}$ is J_j, we will give now existence conditions for having $S(i,j,t) \neq \emptyset$. In this case, a lower bound for tardiness cost incurred by job J_i in any sequence of S(i,j,t) will be calculated.

Job J_i will be called a "predecessor" of job J_j if iPj and no job J_k, distinct from J_i and J_j, satisfies iPk and kPj.

Lemma 1. For any t, $S(i,j,t) = \emptyset$ if

(i) $i \in B_j$ and J_i is not a predecessor of J_j

or (ii) $i \in A_j$.

Proof. (i) a job J_k included between J_i and J_k must be processed between the t^{th} and $(t+1)^{st}$ phase, which is impossible.

(ii) obvious.

#

Defining $B(i,j) = \{i\} \cup B_i \cup B_j$ and denoting by $|B(i,j)|$ the number of its elements, we have:

Lemma 2. $S(i,j,t) \neq \emptyset$ if the following relation:

$$|B(i,j)| \leqslant t \qquad (2\text{-}1)$$

does not hold.

Proof: all the jobs whose indices are in $B(i,j)$ must be processed in the t first phases.

#

If (2-1) holds with equality, the set of the (t+1) first jobs is uniquely determined and:

$$C_{ij}^{t} = C_j\Big(\sum_{k \in B(i,j)} p_k + p_j\Big)$$

is the exact tardiness cost incurred by job J_j.

If (2-1) holds with a strict inequality, then the remaining jobs to be processed before J_i and J_j cannot be chosen among the descendants of J_i and J_j. So, defining $A(i,j) = \{j\} \cup A_i \cup A_j$

and $R(i,j) = \{1,2,..,n\} - (B(i,j) \cup A(i,j))$,

we have the following lemma:

Lemma 3. $S(i,j,t) = \emptyset$ if the following relation:

$$|R(i,j)| \geqslant t - B(i,j) \qquad (2\text{-}2)$$

does not hold.

Proof: obvious from the above arguments.

#

If (2-2) holds with an equality then the set of jobs to be processed in the (t+1) first phases is exactly B(i,j) U R(i,j).

If (2-2) holds with a strict inequality, then a lower bound on the completion time τ_{ij}^t of job J_j is obtained by choosing the $(t - |B(i,j)|)$ jobs in R(i,j) with least processing time. Since $C_j(t)$ is a non decreasing function, we have:

Lemma 4. $C_{ij}^t = C_j(\tau_{ij}^t)$ is a lower bound of the tardiness cost for J_j when J_j is processed in the $(t+1)^{st}$ phase, just after job J_i.

With the lower bound C_{ij}^t of lemma 4, a time-dependant TSP may be used for providing a lower bound on the total tardiness cost: define a multipartite graph like in PART ONE, with dummy initial node (0) and final node (n+1). The arcs $((J_i, t), (J_j, t+1))$ have length C_{ij}^t when these arcs are defined (i.e. when (2-1) and (2-2) hold). The first phase nodes (J_i, t) are defined only for those jobs J_i with $B_i = \emptyset$ and $C_{oi}^o = C_i(p_i)$; the next phase "active" nodes are recursively defined as follows:

"$(J_j, t+1)$ exists if and only if there is a node (J_i, t) and C_{ij}^t is defined."

Last edges $((J_i, n), n+1)$ are dummy and have a length equal to 0.

Illustration: The following example is taken from [18] with

$$C_i(t) = \alpha_i \text{Max } \{0, t-d_i\}.$$

The data are given in Table II

Table II

i	1	2	3	4	5	6	7
P_i	12	13	14	16	26	31	32
d_i	42	33	51	48	63	88	146
α_i	7	9	5	14	10	11	8

The precedence graph is given in Fig. 2

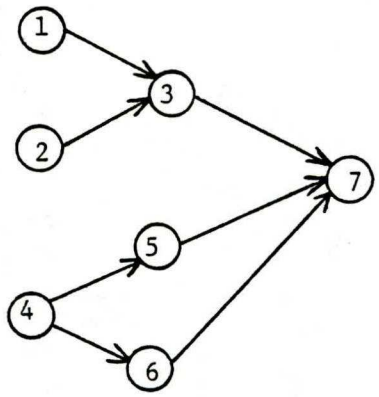


FIGURE 2

The lower bounds C_{ij}^{t} on the costs and the corresponding multipartite network were given to illustrate the PART ONE. Note that the final node corresponds to job J_7 since, from the precedence graph, J_7 must be processed in the last position.

All feasible sequences S (including P*) satisfy:

$$a_i^S = \underline{1} \quad \text{for all } i, \text{ so:}$$

$$\ell(w) \leqslant \ell^S \quad \text{for all sequences } S$$

From (ii) and theorem 1, it follows that:

$$C(w) \leqslant C(S) \quad \text{for all sequences } S.$$

#

If only condition (i) is satisfied, then the corresponding sequence w is a "good" solution to MTCP and the value of an optimal solution S* must satisfy:

$$\ell(w) \leqslant C(S^*) \leqslant C(w)$$

If condition (i) is not satisfied, a stopping criterion, as defined in PART ONE, occurs and the maximum length $\ell(\P^e)$ of a shortest path is a lower bound to C(S*) i.e.

$$\ell(\P^e) \leqslant C(S^*)$$

A Branch and Bound, based on the framework defined in PART ONE, is now required. Before describing its specific details, it is advisable to introduce powerful reductions in the multipartite network.

Theorem 1. If $(J_{w(1)}, J_{w(2)},\ldots,J_{w(n)})$ is a sequence satisfying the given precedence relation, with tardiness cost C(w) and $(0,(J_{w(1)},1), (J_{w(2)},2),\ldots,(J_{w(n)},n), n+1)$ is the corresponding path, with length $\ell(w)$ in the multipartite graph then $\ell(w) \leqslant C(w)$

Proof: the theorem is justified by the above lemmas 1 to 4. #

Since the length of the path associated with a sequence is only a lower bound for the tardiness cost of this sequence, it is not necessary to solve the TDTSP; a procedure providing a good lower bound on its solution is sufficient and this is performed by the subgradient technique described in PART ONE.

Theorem 2. If, for some penalty vector ¶, the shortest path P* satisfies the two following relations:

(i) it contains exactly one node (J_i, t) associated with every job J_i, defining the permutation w (by w(t)=i)

(ii) $\ell(w) = C(w)$

then the sequence defined by w is a minimum cost sequence.

Proof: Since P* is the shortest path for the penalty vector ¶, we have, for all paths P

$$\ell(w) + \sum_i a_i^{P*} \P_i \leqslant \ell^P + \sum_i a_i^P \P_i$$

The reductions will be performed by deleting one of the two arcs defined by the costs C^t_{ij} and C^t_{ji} as some condition holds.

Consider J_i and J_j such that no precedence relation holds between them. Denote by Θ^t_{ij} a lower bound on the total processing time of the (t -1) first jobs in a feasible sequence in which J_i and J_j are processed respectively in t^{th} and $(t + 1)^{th}$ position. Remark that

$$\tau^t_{ij} = \tau^t_{ji} = \Theta^t_{ij} + p_i + p_j$$

Denote also by $\bar{\Theta}^t_{ij}$ an upper bound on the same value (of course $\Theta^t_{ij} = \Theta^t_{ji}$ and $\bar{\Theta}^t_{ij} = \bar{\Theta}^t_{ji}$).

Theorem 3: If, for all $\theta \in \left[\theta^t_{ij}\ ,\ \bar{\theta}^t_{ij}\right]$

we have $C_j\ (\theta+p_i+p_j) - C_j\ (\theta+p_j) \leqslant C_i\ (\theta+p_i+p_j) - C_i(\theta+p_i)$ (2.3)

then the arc defined by the cost C^t_{ji} can be deleted.

Proof: In any feasible sequence with J_j as the t^{th} job and J_i as the $(t+1)^{th}$ job, the completion time of the $(t-1)^{th}$ job is some $\theta \in \left[\theta^t_{ij}, \bar{\theta}^t_{ij}\right]$ (if $t = 1$, the interval is reduced to the single point 0). Then, the improvement obtained by exchanging J_i and J_j is

$(C_i(\theta+p_i) + C_j(\theta+p_i + p_j)) - (C_j(\theta+p_j) + C_i(\theta+p_i+p_j)) \leqslant 0.$

Thus, any feasible sequence in which J_j is the t^{th} job and J_i the $(t+1)^{st}$ may be improved by this exchange. #

Of course, if (2-3) holds with equality, only one of the two arcs defined by C^t_{ij} and C^t_{ji} is deleted. The power of these reductions will be illustrated for the example in section 2-4.

The branch and bound structure is based on the framework defined in PART ONE. The main difference comes from the fact that, for any sequence, its length in the network is only a lower bound on its cost. When a shortest path occurs to be a sequence-path, its length is the best possible bound obtainable in the corresponding problem. It is necessary to compute the cost of this sequence: if this cost equals the length of the path, then the optimal sequence is obtained for the problem, and backtracking is performed.

But, in the other case, when the cost is greater than the length of the path, it is necessary to branch from this problem for further tightening of the bound.

The dominance test may also be performed in a more efficient way. Indeed, the cost of assigning the job J_r in the k^{th} position does not depend on which job is processed just before, but only on the total processing time of the (k-1) first jobs. So the insertions may be checked with J_r as first job in the segment.

2.4 The Weighted Tardiness problem.

For the application to the weighted tardiness problem, in which $C_i(t) = \alpha_i \text{ Max } \{0; t-d_i\}$, some additional specifications will be used.

The branch and bound algorithm will make use of the Elmaghraby test (i.e if the due date of a job J_i is not less than the total processing time, then process J_i as the last job) as first operation when defining any subproblem.

The precedence relation is not recomputed, but only copied from the one in the original problem. The reduced multipartite network is then defined from the precedence relation.

The lower bounds θ_{ij}^t (providing τ_{ij}^t) are obtained by simply selecting the jobs in R(i,j) with least processing time. In the same way, the upper bounds $\bar{\theta}_{ij}^t$ are obtained by selecting the jobs in R(i,j) with greatest processing time.

For the reductions, two more precise statements of theorem 3 hold.

Corollary 1. If (i) $d_i \leq d_j$

and (ii) $\bar{\theta}_{ij}^t \leq \text{Min } (d_i, d_j - p_j) - p_i$

then delete the arc defined by C_{ji}^t.

Proof: Condition (ii) implies $\theta + p_i \leq \bar{\theta}_{ij}^t + p_i \leq d_i$

and $\theta + p_i + p_j \leq \bar{\theta}_{ij}^t + p_i + p_j \leq d_j$

so the two jobs are processed before their due-date if this is done according the EDD (earliest due-date) order defined by condition (i). #

Corollary 2. If (i) $p_i/w_i \leq p_j/w_j$

and (ii) $\theta_{ij}^t \geq \text{Max } (d_i - p_i, d_j - p_j)$

then delete the arc defined by C_{ji}^t.

Proof: Condition (ii) implies

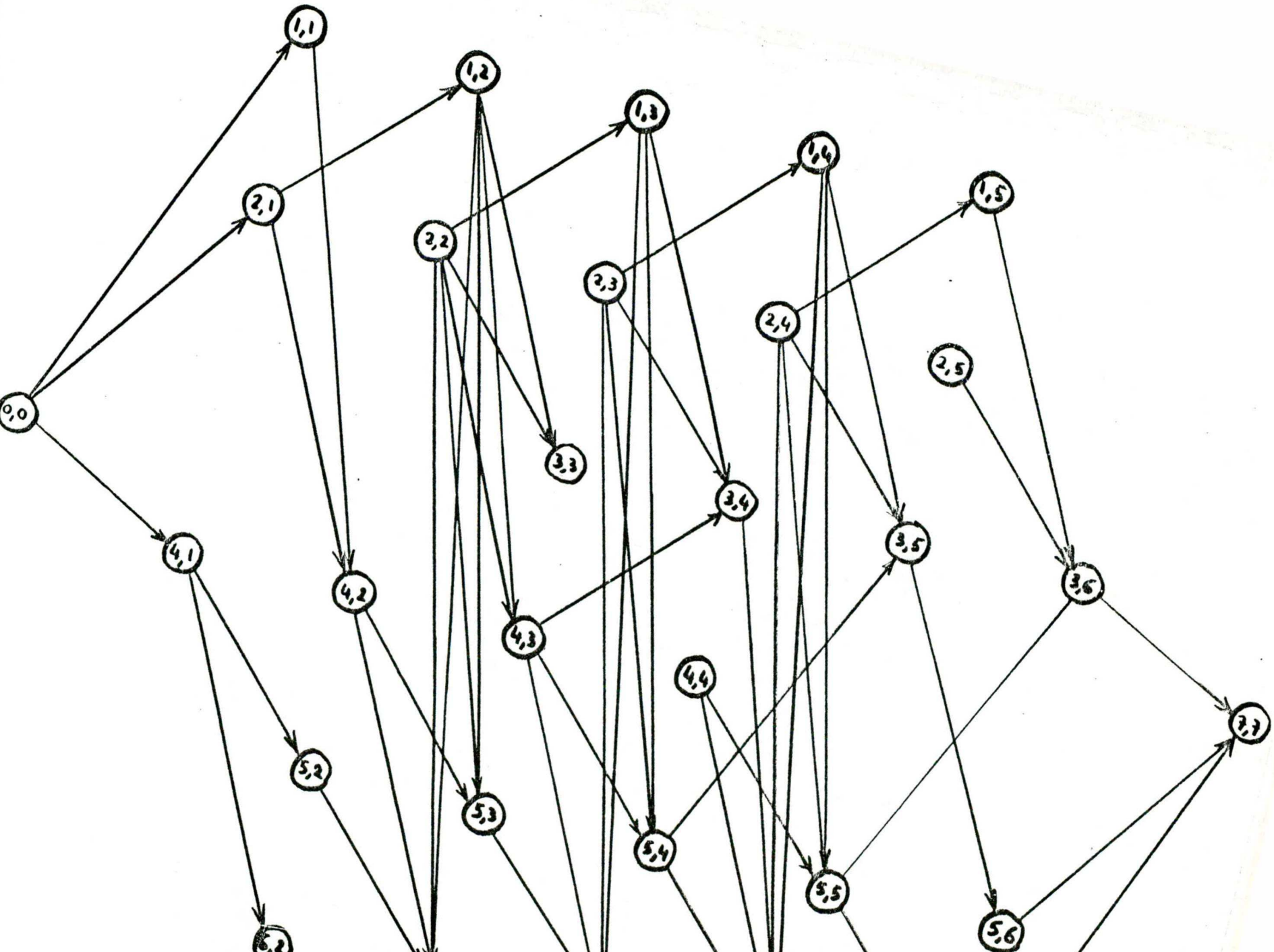
$\theta + p_i \geq d_i$ and $\theta + p_j \geq d_j$

So J_i and J_j are late jobs as soon as processed in t^{th} position. Then the WSPT order is the best one, since the costs are reduced to weighted lateness. #

The implementation of these tests is performed for each pair of incomparable (J_i, J_j) by successively computing θ_{ij}^t (used for defining τ_{ij}^t) for $t = |B(i,j)|$ until $t = |B(i,j)| + |R(i,j)|$. The computation of θ_{ij}^t is abandonned as soon as Corollary 1 does not apply.

Illustration: if Corollaries I and II are applied to the illustrative 7-job example. One obtains the reduced network pictured in Fig. 3. Note that several arcs and nodes become meaningless: for example, node (2,2) and the issued arcs may be deleted (no path from the origin to the end may use them). After the deletion of useless elements, the network is then reduced to the one pictured in Fig.4. Clearly the problem is then solved in one (trivial) shortest path iteration.

Finally the dominance test, as described in the previous section, is applied to each job in the last phase. At each improvement, the corresponding job is erased from the last phase. If after these tests, there is no job in the last phase, then backtracking occurs; if there is just one job, then a simple branching is decided, otherwise shortest path iterations are performed, as described in PART ONE.


1,1
1,2
1,3
1,4
1,5
2,1
2,2
2,3
2,4
2,5
0,0
3,3
3,4
3,5
3,6
4,1
4,2
4,3
4,4
3,7
5,2
5,3
5,4
5,5
5,6

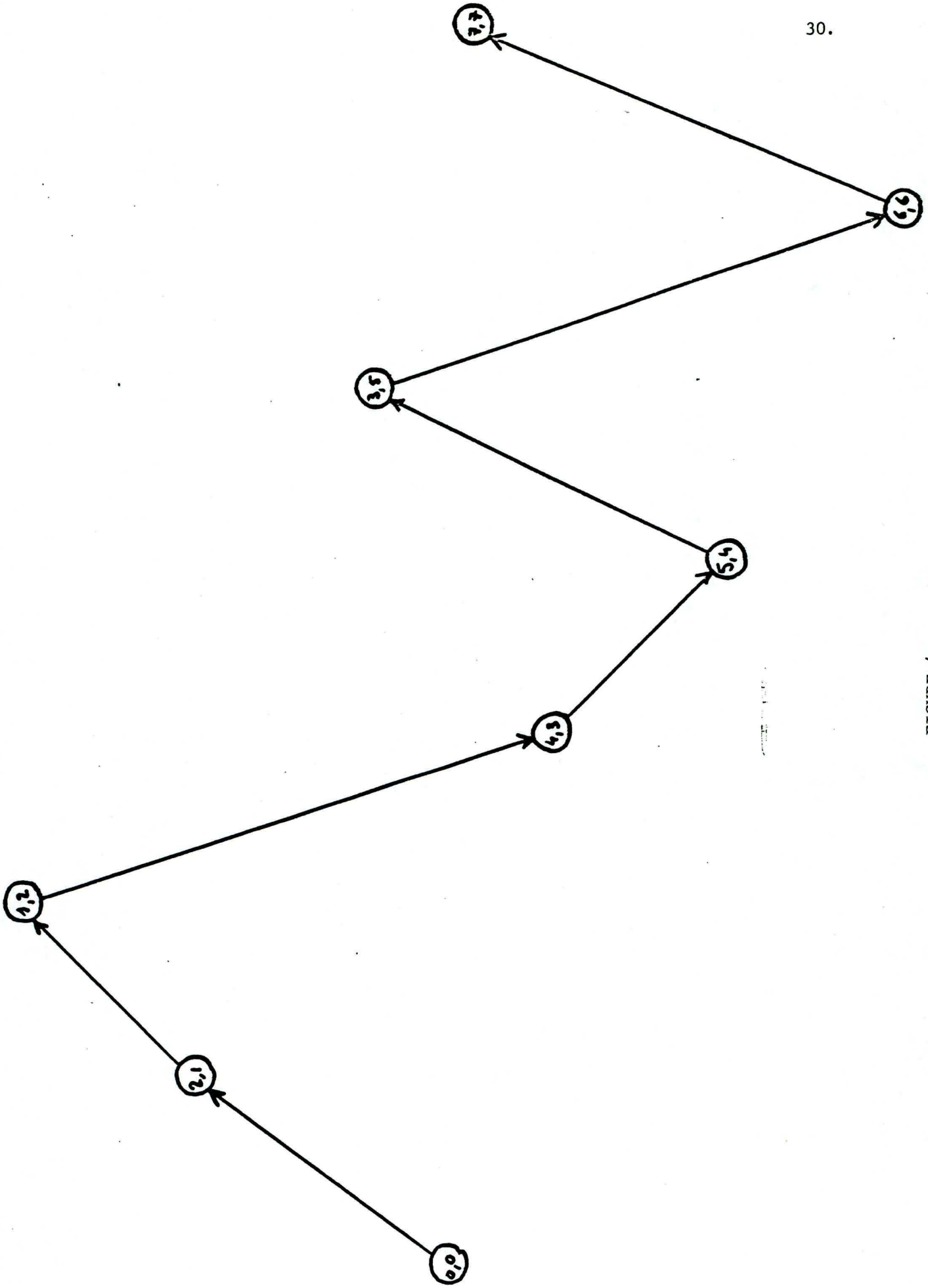


FIGURE 4

2.4 Computational Results

The algorithm was coded in FORTRAN for the Weighted Tardiness problem and run on the CDC Cyber 74-18 of the University of Montreal on 84 test problems, with 15 and 20 jobs. The data for the first set of 42 problems are those of Rinnooy Kan and al.[18], and tests were performed on the 15-job problems with tardiness factor t of 0.6 and 0.8, and the 20-job problems with t = 0.4, 0.6 and 0.8. The solution time and number of nodes in the enumeration are summarized in Tables III and IV.

The mean solution times (resp. number of nodes) are given for each set of problems with fixed size and tardiness factor. In order to provide comparisons with the algorithm of Rinnooy Kan and al., the median and maximum for the solution time and the number of nodes are also given.

The second set of 42 problems adresses the Total Tardiness problem, and was solved with the same algorithm. While some simplifications are possible in the case where $\alpha_i=1$ for all i, for instance a reduction of about one third in the execution time of the heuristic, no such modification was attempted in order to study the behaviour of our algorithm for this special case. The test data are those of the first set, except that the weights are uniformly set to 1. The mean and maximum of the solution time and the number of nodes are given in table V.

Considering that the CYBER 74 is reported to be from 2.5 to 3.5 faster than the CYBER 73 used by Rinnooy Kan & al. (hereafter RKLL), the results given in Table III show a real improvement.

It appears that the improvement over RKKL is about of the same order of magnitude as the improvement of RKLL over the previous existing methods. The comparisons for small n and t have less meaning due to the low computation time involved, and the fact that our algorithm is preceded by the heuristic method which requires a fairly constant time of 0.5 second for n = 15 and 1.1 seconds for n = 20.

For the total tardiness problem, no direct comparison with the work of Fisher[6] is feasible. Fisher reports in[6] some comparison between his algorithm running on IBM 360 - 67 and RKLL on CDC Cyber 73-38 and comments: "generally their solution times were about equal to ours (Fisher) for the 20-job problems". It seems that, at least for the 20-job problems, our algorithm will perform better than Fisher's. However, further computational tests are forecast, as soon as the corresponding data are available to the authors.

Some improvements to the existing algorithm are possible. First, attempts to refine on the precedence relation should be performed at each node in the enumeration. The arc length C_{ij}^{t} in the multipartite network should provide better bounds if the value of θ_{ij}^{t} were computed in a more accurate way than by just considering the jobs with least processing time in R(i,j). For instance, for $t = |B(i,j)| + 1$, it is more advisable to select, among the jobs in R(i,j) which are preceded by no other job in R(i,j), the one with least processing time. For greater values of t, this selection turns out to become a very difficult problem in itself, but it is likely that tigher bounds could be obtained at a small additional expense. Reductions using Theorem 3

could be extended for the Weighted Tardiness problem to cases in which Corollaries 1 and 2 do not apply, that is in the "middle part" of the network. Finally the shortest path algorithm could be replaced by an algorithm which would reject paths including segments such as (i,t), $(j,t+1)$, $(i,t+2)$. This shortest path algorithm is described in [16], and works in an average running time less than twice the time of the simple dynamic programming algorithm.

Besides these attempts to refine the lower bounds, the branch and bound structure itself could be improved. Considering that the costs C_{ij}^{n-1} in the last transition are exact, and the best order of the two last jobs is simply determined, one could extend the implicit bounds and the branching scheme to pairs of jobs in the two last phases. So the branch and bound will work implicitly two levels down. Of course, this may be extended to triples of jobs in the three last phases, and generally to r-tuples of jobs in the r last phases. It is likely that there is a threshold value for r, from which it becomes much too difficult to handle the r-tuples for larger values of r. However, it is not obvious that the best value of r is for r=1, which is currently used in our algorithm.

The given algorithm may be simply adapted to handle more difficult problems than the tardiness problem. Set-up costs d_{ij}, occuring when job J_i is followed by job J_j on the machine, are easily handled in this model, by simply adding d_{ij} to the arc length C_{ij}^t, for all t such that C_{ij}^t is defined. More generally, these set-up costs may be time-dependent exactly in the sense described in PART ONE. It should be noted that the bounds C_{ij}^t become tighter as the set-up part of the costs are more important.

It appears that the model presented here is able to handle one-machine problems with more general cost functions than any existing algorithm.

TABLE III

SOLUTION TIME (CPU Seconds)

n	t	Number of problems	New algorithm*			Rinnooy Kan Lageweg Lenstra$	
			Average	Median	Maximum	Median	Maximum
15	0.6	12	1.9	1.6	5.9	6.3	121.8
	0.8	12	1.7	1.6	3.4	45.6	85.6
20	0.4	6	2.7	1.5	6.4	1.1	20.3
	0.6	6	13.6	6.5	30.7	180.8	>300
	0.8	6	12	11.0	20.8	>300	>300

* CDC Cyber 74-18

$ CDC Cyber 73-28

TABLE IV

Number of Nodes

n	t	Number of problems	New algorithm			Rinnooy Kan Lageweg Lenstra	
			Average	Median	Maximum	Median	Maximum
15	0.6	12	29.5	17	121	647	9564
	0.8	12	28.0	24	57	4532	9952
20	0.4	6	11.3	1	34	25	1206
	0.6	6	118.8	48	302	11105	--
	0.8	6	136.6	120	327	--	--

TABLE V

Computational Results with $\alpha_i=1$

n	t	Number of problems	Solution Time * (CPU seconds)		Number of Nodes	
			Average	Maximum	Average	Maximum
15	0.6	12	0.8	1.0	9.0	15
	0.8	12	0.7	1.0	12.4	43
20	0.4	6	1.1	1.6	5.5	14
	0.6	6	2.7	5.7	21.0	52
	0.8	6	3.7	12.8	38.8	168

* CDC Cyber 74-18

EXTENSIONS AND CONCLUSION

The TDTSP which was described in PART ONE and applied to the Tardiness Cost problem is PART TWO, may be used to approach several optimum permutation and related problems.

The first obvious application is to the Traveling Salesman Problem. Actually, this provided the basic impetus for developping the general model. The first work in this area was described in two earlier research reports [15], [16]. D. Houck and R.Vemuganti:, from Baltimore, have independently discovered the same approach to the TSP and provided some experimental comparisons with the "1-arborescence" approach suggested by Held and Karp [9], for the asymetric problem [12]. A joint publication is currently in preparation. The approach may be extended to several routing problems, for instance to the multiple TSP in which the salesmen have to visit the same number of customers. The reader is referred to [16] for further description of the routing applications.

Some extensions of the scheduling applications for the one-machine problem were outlined at the end of PART TWO. It appears that the method could be extended to problems involving several machines, such as flow-shop and job-shop problems, and this would require further examination.

Another optimum permutation problem, somewhat related to one-machine scheduling problems, is the "Linear Ordering" (D.Adolphson and T.C. Hu [1]). This problem may be handled as the Weighted Tardiness problem described in PART TWO. The main difference lies in the way the bounds defining the multipartite network are computed.

Turning to a more general formulation of optimum permutation problems, one obtains the "Quadratic Assignment problem", see [14] for instance. This problem appears to be a very general way to formulate several optimization problems, and turns out to be a very difficult problem. At the present time exact solution methods are tractable only for small-sized problems. In the application of the TDTSP model, the computation of the bounds used to defined the multipartite network is performed through the solution of related assignment problems. These last two applications (linear ordering and quadratic assignment problem) are currently studied by the authors.

It appears that much work has to be done, as well for improving the present approach as for applying it to several existing difficult problems.

ACKNOWLEDGEMENTS

The authors would like to thank Jean-Claude Nadeau and Ben T. Smith for their helpful assistance. This research was supported by The National Research Council of Canada, Grants CNRC A8528 and RD804.

REFERENCES

[1] D. Adolphson and T.C. Hu, "Optimal Linear Ordering", SIAM J. Appl.Math.25, 403-423 (1973).

[2] K.R. Baker, "Introduction to Sequencing and Scheduling", John Wiley and Sons, New York (1974).

[3] K.R. Baker and J.B. Martin, "An experimental Comparison of Solution Algorithms for the Single-Machine Tardiness Problem", Naval Res.Log. Quart. 21, 187-199 (1974).

[4] R.W. Conway, W.L. Maxwell and L.W. Miller, "Theory of Scheduling", Addison-Wesley, Reading, Mass. (1967).

[5] H. Emmons, "One-Machine Sequencing to Minimize Certain Functions of Job Tardiness", Opns.Res. 17, 701-715 (1969).

[6] M.L. Fisher, "A Dual Algorithm for the One-Machine Scheduling Problem", to appear in Mathematical Programming.

[7] K.R. Fox, "Production Scheduling on Parallel Lines with Dependencies", Ph.D. Dissertation, The Johns Hopkins University, Baltimore, Md. (1973).

[8] L. Gelders and P.R. Kleindorfer, "Coordinating Aggregate and Detailed Scheduling Decisions in the One-Machine Job Shop: Part I. Theory", Opns.Res. 22, 46-60 (1974).

[9] M. Held and R.M. Karp, "The Traveling-Salesman and Minimum Spanning Trees", Opns Res. 18, 1138-1162 (1970).

[10] M. Held and R.M. Karp, "The Traveling-Salesman and Minimum Spanning Trees", Part II, Math.Programming 1, 6-25 (1971).

[11] M. Held, P. Wolfe H. Crowder, "Validation of Subgradient Optimization", Math.Programming 6, 62-88 (1974).

[12] D.J. Houck and R.R. Vemuganti, "The Traveling Salesman Problem and Shortest n-Paths", presented at ORSA-TIMS Meeting, Philadelphia, Pa (1976).

[13] R.M. Karp "Reducibility among Combinatorial Problems", pp. 85-103 in "Complexity of Computer Computations", (R.E. Miller and J.W. Thatcher, eds.), Plenum Press, New York (1972).

[14] M.Los, "Experimental Comparison and Evaluation of Several Exact and Heuristic Algorithms to Solve Quadratic Assignment Problems of the Koopmans-Beckmann Type", Centre de Recherches sur les Transports, Université de Montréal (1976).

[15] J.C. Picard and M. Queyranne, "Le Problème du Voyageur de Commerce: une Formulation par la Programmation Linéaire" Rapport Technique EP75-7, Ecole Polytechnique, Montréal (1975).

[16] J.C. Picard and M. Queyranne, "Le Problème du Voyageur de Commerce: Plus-Courts Chemins et Optimisation par Sous-Gradients", Rapport Techn.EP76-7, Ecole Polytechnique, Montréal (1976).

[17] S.Reiter and G. Sherman, "Discrete Optimizing", SIAM J. 13, 864-889 (1965).

[18] A.H.G. Rinnooy Kan B.J. Lageweg and J.K. Lenstra, "Minimizing Total Costs in One-Machine Scheduling", OpnsRes.23, 908-927 (1975).

[19] B. Roy, "Algebre Moderne et Théorie des Graphes", Dunod, Paris, (1970).