DÉPARTEMENT DE GÉNIE INDUSTRIEL

Rapport Technique EP77-R-12
Classification: Library of Congress No.....

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

by

Jean-Claude Picard

and

Maurice Queyranne

Revised version, February 28th, 1977

Ecole Polytechnique de Montréal

Campus de l'Université
de Montréal
Case postale 6079
Succursale 'A'
Montréal, Québec
H3C 3A7

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

by

Jean-Claude Picard

Department of Systems Engineering
Federal University of Paraiba
Campina Grande, Brasil

and

Maurice Queyranne

Département de Génie Industriel
Ecole Polytechnique
de
Montréal

Revised version,... February 28th, 1977

ABSTRACT:

The time-dependent traveling salesman problem may be stated as a scheduling problem in which n jobs have to be processed at minimum cost on a single machine. The set-up cost associated with each job depends not only on the job which precedes it, but also on its position (time) in the sequence. The optimization method described here combines finding shortest paths in an associated multipartite network with subgradient optimization and some branch and bound 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 attacked by this method. A branch and bound algorithm is designed for this problem; it uses a related time-dependent traveling salesman problem to compute the required lower bounds. Computational results are given for the weighted tardiness problem.

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 cost of each transition depends not only on the two respective locations involved, but also on their positions in the sequence which defines the tour.

In Section 1 are given three integer programming formulations of the TDTSP, distinct from those previously given by K. Fox [10]. The relations between relaxations of these three problems are then studied. Comparing the lower bounds provided by these relaxations, it is shown that the shortest path relaxation is equivalent to an extension of Hadley's TSP formulation (see [16]), and better than an assignment relaxation. Consequently, the basic method used here seeks a shortest path with specified characteristics in an associated multipartite network, using penalties to enforce the additional constraints. Since a duality gap may exist a branch and bound algorithm is constructed. A general and powerful dominance test is introduced to reduce the enumeration effort.

In Section 2, general one-machine sequencing problems are shown to be related to the TDTSP. The techniques of Section 1 are then applied to solve the class of such problems which consists of minimizing a cost which is a non-decreasing function of the completion time of each job. This is done by defining a multipartite network, in which shortest paths are used as in Section 1 for providing lower bounds on the optimal cost. Efficient reductions in this network are described. This bounding scheme and the dominance test are inbedded in a branch and bound algorithm. The computational results given indicate that this algorithm is an efficient one for the weighted tardiness problem in sequencing jobs on one machine.

A final section on extensions and conclusions provides background material and outlines the application of the TDTSP methodology to several classes of problems in scheduling, routing and location theories.

1. THE TIME-DEPENDENT TRAVELING SALESMAN PROBLEM.

Definition:

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

Consider a set of n jobs, denoted by $J_1,\ldots,J_n$, to be performed on a single machine and set-up costs C_{ij}^t occurring when job J_i, processed in the t^{th} position in the sequence, is followed by 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 begins. It has to be returned to a given "final state", denoted by (n + 1), after the job processing ends, and initial and final set-up costs $C_{0\ i}^0$ 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

$$C_{(w)} = C_{0\ w(1)}^0 + C_{w(1)\ w(2)}^1 + \ldots + C_{w(n-1)\ w(n)}^{n-1} + C_{w(n)\ \ n+1}^n \qquad (1\text{-}1)$$

Example 1:

n = 4 and the set-up costs are given in Table 1.

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

$$C_{(w)} = C_{0\ 1}^0 + C_{1\ 2}^1 + C_{2\ 3}^2 + C_{3\ 4}^3 + C_{4\ 5}^4$$

$$= 33 + 2 + 6 + 15 + 22 = 78.$$

Remarks:

1 - The diagonal entries $C_{i\ i}^t$ have no meaning and therefore need not be defined.

2- Problems with unspecified initial (final) state can be formulated in the same way, using $C_{0i}^0 = 0 (C_{i\ n+1}^n = 0)$ for all i.

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

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

for all (i,j) the problem reduces to the classical traveling salesman problem (TSP).

4- When the set-up costs are not dependent on the second job of the pair, that is, when

$$c_{ij}^{t} = c_{i}^{t}, \qquad \text{all } j,$$

for all (i,t), the problem reduces to the classical assignment problem (AP). By symmetry, this is also true when the set-up costs are not dependent on the first job of the pair.

The TDTSP was defined by Kenneth R. Fox in his dissertation [10], and it was illustrated with examples from the brewing industry. The following one is a variation on a classical illustration ([7, p.53]) of the TSP.

Illustration: Consider a paint factory which produces 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.

Integer Programming Formulations:

In his thesis, [10] K. Fox gives five linear integer programming formulations, and one flow-with-gains formulation for the TDTSP.
None of these formulations was found to lead to a tractable solution scheme, and Fox reports his branch and bound algorithm being unable to solve a 10-job problem within 12 minutes (computer not specified). We propose here three other integer programming formulations and will study the relations between relaxations of these problems. Our method for solving the TDTSP uses two of these formulations.

Sequence position 0

C^0_{0j}	1	2	3	4
0	33	30	36	6

Sequence position 1

C^1_{ij}	1	2	3	4
1		2	26	0
2	46		30	11
3	28	15		8
4	15	0	10	

Sequence position 2

C^2_{ij}	1	2	3	4
1		41	25	15
2	1		6	38
3	8	26		23
4	10	16	0	

Sequence position 3

C^3_{ij}	1	2	3	4
1		41	8	30
2	3		9	12
3	14	49		15
4	39	41	39	

Sequence position 4

C^4_{i5}	5
1	30
2	46
3	40
4	22

Table 1. The set-up costs C^t_{ij} for example 1.

The first integer programming formulation of the TDTSP is a straightforward generalization of the TSP formulation given by Hadley and simplified by Houck and Vemuganti [16], in which the costs (distances) have simply been made time-dependent:

(ILP1)

$$\text{Min} \sum_{j=1}^{n} C_{0j}^{0} x_{0j}^{0} + \sum_{t=1}^{n-1} \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j\neq i}}^{n} C_{ij}^{t} x_{ij}^{t} + \sum_{j=1}^{n} C_{j\,n+1}^{n} x_{j\,n+1}^{n}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} x_{0j}^{0} = 1$$

$$x_{0j}^{0} = \sum_{\substack{k=1 \\ k\neq j}}^{n} x_{jk}^{1} \qquad j = 1,\ldots,n$$

$$\sum_{\substack{i=1 \\ i\neq j}}^{n} x_{ij}^{t} = \sum_{\substack{k=1 \\ k\neq j}}^{n} x_{jk}^{t+1} \qquad t = 1,\ldots,n-2; \quad j = 1,\ldots,n$$

$$\sum_{\substack{i=1 \\ i\neq j}}^{n} x_{ij}^{n-1} = x_{jn+1}^{n} \qquad j = 1,\ldots,n$$

$$x_{0j}^{0} + \sum_{\substack{i=1 \\ i\neq j}}^{n} \sum_{t=1}^{n-1} x_{ij}^{t} = 1 \qquad j = 1,\ldots,n$$

$$x \geqslant 0$$

$$x \text{ integer}$$

Note that the above formulation is distinct from the five integer programming formulations of Fox, which all include a constraint with coefficients which are not in the set $\{0,1,-1\}$, as is the case for (ILP1).

The second formulation refers to the Quadratic Assignment Problem (QAP) (cf [20], [21]) which is defined as follows:

(QAP)

$$\text{Min} \quad \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{p=1}^{n} \sum_{q=1}^{n} a_{ijpq} x_{ip} x_{jq}$$

$$\text{s.t.} \quad \sum_{p=1}^{n} x_{ip} = 1 \qquad i = 1\ldots,n$$

$$\sum_{i=1}^{n} x_{ip} = 1 \qquad p = 1\ldots,n$$

$$x_{ip} = 0 \text{ or } 1 \qquad \text{all } i,p$$

The TDTSP may be formulated as a QAP by defining $a_{ijpq} = 0$ except for:

$$a_{iill} = C^{0}_{0i} \qquad \text{all } i$$

$$a_{ijtt+1} = C^{t}_{ij} \qquad \text{all } i,j,t \quad (i \neq j \text{ and } 1 \leqslant t \leqslant n-1)$$

and $a_{iinn} = C^{n}_{i,n+1}$

Before describing the third integer programming formulation, it is necessary to introduce some definitions.

Multipartite Networks

A multipartite graph (cf[24]) is a directed graph (V,A), the vertex set V of which is partitionned into k subsets $V_1, V_2 \ldots, V_k$ (here called phases) such that any arc in A having its origin in a subset V_i has its end in V_{i+1}. A bipartite graph is a multipartite graph with k=2. A multipartite network N = (V,A,C) is a multipartite graph with a weight function C defined on A.

With the TDTSP is associated a multipartite network N = (V,A,C) which forms the basis of the approach to be discussed here. Its vertex set V consists of vertices (0),
(i,t) for i,t = 1,2,...,n, and
(n+1),
where vertex (i,t) represents the possible execution of job J_i in phase(or sequence position)t.

The arcs in A are:
initial arcs (0,(i,1)) with length $C^{0}_{0\ i}$,

transition arcs ((i,t),(j,t+1)) with length $C^{t}_{i\ j}$ $(i \neq j)$,
final arcs ((i,n), n+1) with length $C^{n}_{i\ n+1}$.

The multipartite graph associated with the 4-job TDTSP of example 1 is pictured in Fig. 1.
A path P in N joins 0 to (n+1) through n transition vertices, one in each phase.

If every job appears exactly once among the n transition vertices in a path P, then P represents a feasible sequence w, and its length $\ell(P)$ is the cost $C_{(w)}$ of this sequence. Such a path will be called a sequence path.

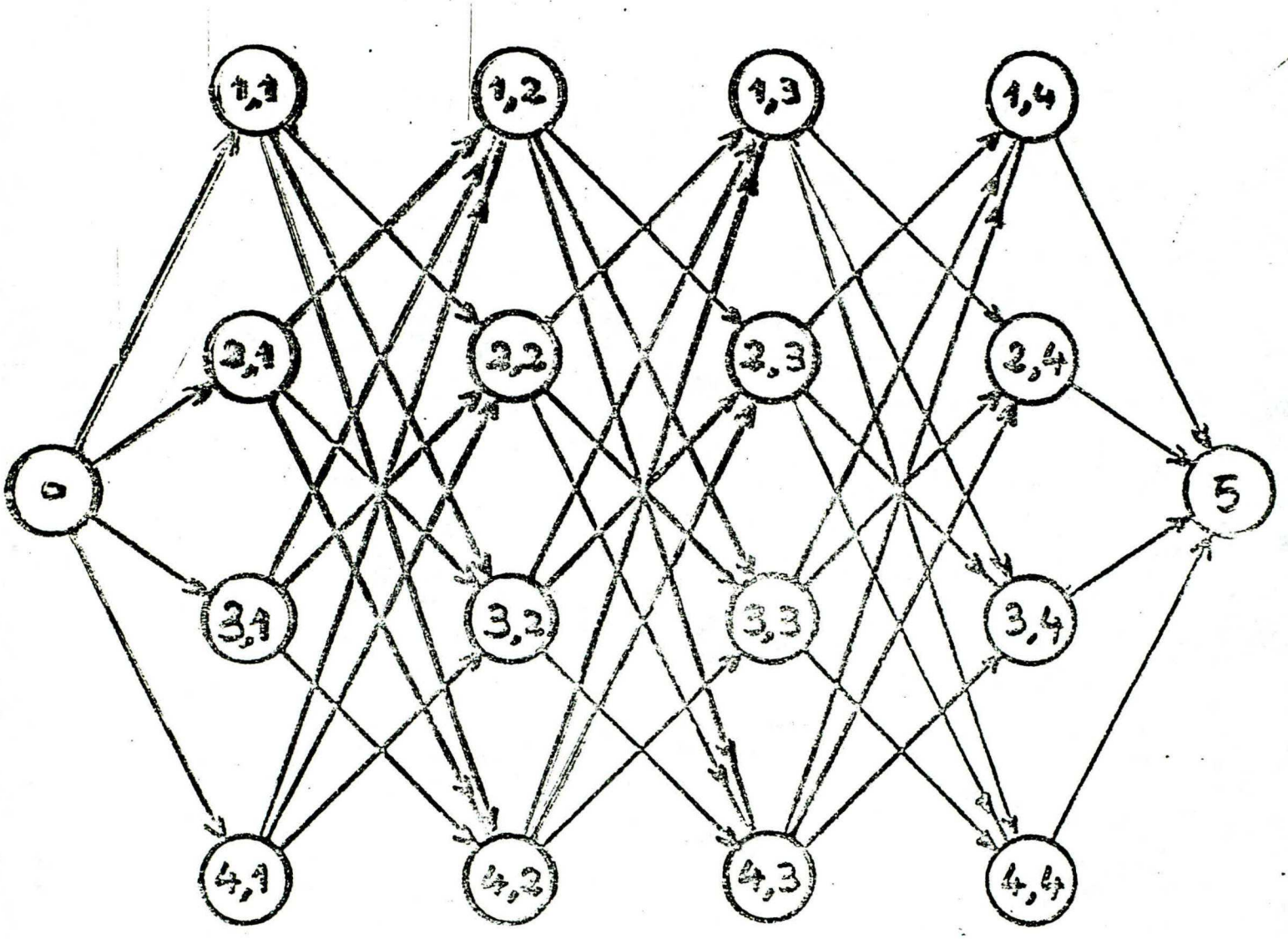


FIGURE 1

Denote by a_i^p the number of occurences of vertices (i,t) (for t=1,...,n) representing job J_i in a path P and let a^p be the corresponding vector, with n components a_i^p (i=1,...,n). Clearly, a path P is a sequence path if and only if $a^p = \underline{1}$, where $\underline{1}$ denotes the vector with all components equal to one. Since a path P contains exactly n jobs, the following holds:

$$\sum_{i=1}^{n} a_i^p = n \qquad (1\text{-}2)$$

Shortest paths:

Finding a shortest path in N may be performed by the following algorithm, which is a straightforward application of dynamic programming:

Shortest-path algorithm:

Step 1 for j=1,...,n set $f(j,1) = C_{0j}^0$

Step 2 for t=1,...,n-1 carry out the step 2a

step 2a for j=1,...,n set

$f(j,t+1) = \text{Min } \{f(i,t) + C_{ij}^t \mid i=1,...,n\}$

and

$m(j,t+1) = k$ such that $f(j,t+1) = f(k,t) + C_{kj}^t$

Step 3 set $\ell = \text{Min } \{f(j,n) + C_{j\ n+1}^n \mid j=1,..,n\}$

and $m(n+1, n+1) = s$ such that $\ell = f(s,n) + C_{s\ n+1}^n$

After completion of this algorithm, ℓ is the length of the shortest-path in N and such a shortest path may be found by backtracking using the labels m. The time-complexity of this algorithm is $O(n^3)$.

As an example, the above algorithm, applied to example 1, yields the path P= (0,(4,1), (2,2), (3,3), (4,4),5) or, in a simpler notation w = (4,2,3,4) with length 49. Since job 1 is omitted and job 4 is carried out twice, this does not define a sequence path.

If the shortest path in N is a sequence path, the corresponding sequence is clearly an optimal solution of the TDTSP. Otherwise, the length ℓ of the shortest path is a lower bound on the cost of an optimal TDTSP sequence. At this point, a naive approach would use a straightforward branch and bound algorithm, fixing and forbidding at certain phases a job which appear more than once. Instead of this, we prefer trying to improve the lower bound before any separation. The motivation and limits of such an attempt will be drawn from the following developments.

Another Integer Programming Formulation

In the following integer linear program, one variable x_p is associated with each path P in N.

$$\text{(ILP2)} \quad \left[\begin{array}{ll} \text{Min} & \sum_P \ell^P x_p \\ \text{s.t.} & \sum_P a_i^P x_p = 1 \qquad i=1,\ldots,n \\ & x_p \geqslant 0, \text{ integer} \end{array}\right.$$

Summing the n constraints, one obtains, after division by n:

$$\sum_P x_p = 1 \qquad (1\text{-}3)$$

Thus an integer solution to ILP2 has exactly one variable equal to one, the others being at zero, and the corresponding path P must be a sequence-path.

Hence, (ILP2) is a valid formulation for the TDTSP.

Relaxations:

Consider a mathematical programming problem

$$\text{(MP)} \quad \left[\begin{array}{l} \text{Min } f(x) \\ \text{s.t. } x \in X \end{array}\right.$$

A problem

$$\text{(MP')} \quad \left[\begin{array}{l} \text{Min } f'(x) \\ \text{s.t. } x \in X' \end{array}\right.$$

is called a relaxation of (MP), according to [12], if and only if $X \subset X'$ and $f'(x) \leqslant f(x)$ for all $x \in X$

There are three types of relaxations to be distinguised:

type 1: $X = X'$ (T1)

type 2: $f'(x) = f(x)$ for all $x \in X$ (T2)

type 3: neither (T1) nor (T2)

In this paper, relaxations of these three types will be met. The implications of this taxonomy will be seen in the difference between the fathoming tests used in the several branch and bound algorithms that can be devised for this type of problems.

Consider first the linear programs (LP1) and (LP2) obtained from (ILP1) and (ILP2) by relaxing the integrality constraints. These are relaxations of type 2. Denote by z_{LP1} and z_{LP2} the optimum values of the objective functions in (LP1) and (LP2), respectively.

Turning to the QAP formulation, consider the linear assignment relaxation, as defined by Lawler [18].

$$
(\text{LAR}) \quad \left[\begin{array}{lll}
\text{Min} & \sum_j \sum_q b_{jq} u_{jq} & \\
\text{s.t.} & \sum_q u_{jq} = 1 & j = 1,\ldots,n \\
 & \sum_j u_{jq} = 1 & q = 1,\ldots,n \\
 & u_{jq} \geqslant 0 & \text{all } j,q.
\end{array} \right.
$$

with $b_{jq} = a_{jjqq} + U_{jq}$, where U_{jq} is the optimal value of the following linear assignment problem:

$$
(\text{LAP}_{j,q}) \quad \left[\begin{array}{lll}
\text{Min} & \sum_{i \neq j} \sum_{p \neq q} a_{ijpq} v_{ip} & \\
\text{s.t.} & \sum_{p \neq q} v_{ip} = 1 & \text{all } i \neq j \\
 & \sum_{i \neq j} v_{ip} = 1 & \text{all } p \neq q \\
 & v_{ip} \geqslant 0 & \text{all } i \neq j, \text{ all } p \neq q.
\end{array} \right.
$$

Since a_{ijpq} is zero when $p \neq q - 1$, (assuming $p \neq q$), the computation of U_{jq} is reduced, for $q \geqslant 2$, to the selection of the smallest $a_{i\,j\,q-1\,q}$ for $i \neq j$. If $q = 1$, then $U_{jq} = 0$.

Note that LAR is a relaxation of type one. Indeed, it is well-known that (LAR) has an optimum solution $\hat{u}$ which is integer, and $\hat{u}$ induces a sequence $\hat{w}$ by defining $\hat{w}(t) = i$ if and only if $u_{it} = 1$.

Denote by z_{LAR} the optimal value of the objective function in LAR.

Theorem 1: Let C* denote the cost of an optimum TDTSP sequence. Then

$$C^* \geqslant Z_{LP1} = Z_{LP2} \geqslant Z_{LAR}$$

Proof: The first inequality follows from the fact that (LP1) is a relaxation of (ILP1).

The following equality is the time-dependent version of a theorem by Houck and Vemuganti [16]. Its validity for the time-dependent case is a straightforward extension of their result.

The last inequality may be obtained by proving that any feasible solution $\bar{x}$ to (LP1) induces a feasible solution $\bar{u}$ to (LAR), defined by

$$\bar{u}_{j1} = \bar{x}^{o}_{oj} \qquad \text{all } j$$

$$\bar{u}_{jq} = \sum_{i \neq j} \bar{x}^{q-1}_{ij} \qquad \text{all } j, \text{ all } q \geqslant 2$$

From the definition of the b_{jq}, one finds

$$b_{j1} = a_{jj11} = c^{0}_{0j} \qquad \text{all } j$$

$$b_{jq} = \text{Min } \{a_{ijq-1} \mid i \neq j\} = \text{Min } \{c^{q-1}_{ij} \mid i \neq j\}$$

$$\text{all } j, \text{ all } q \text{ with } 2 \leqslant q \leqslant n-1$$

$$b_{jn} = a_{jjnn} + \text{Min } \{a_{ijn-1n} \mid i \neq j\} = C^{n}_{j\ n+1} + \text{Min } \{C^{n-1}_{ij} \mid i \neq j\}$$

Hence,

$$\sum_{j,q} b_{jq} \bar{u}_{jq} \leqslant \sum_{j} C^{o}_{oj} \bar{x}^{o}_{oj} + \sum_{\substack{t,i,j \\ i \neq j}} C^{t}_{ij} \bar{x}^{t}_{ij} + \sum_{j} C^{n}_{j\ n+1} \bar{x}^{n}_{jn+1} \qquad (1\text{-}4)$$

Since $\bar{u}$ is feasible for (LAR), then $z_{LAR} \leqslant \sum_{j,q} b_{jq} \bar{u}_{jq}$ and the required inequality is proven by taking as $\bar{x}$ an optimal solution to (LP1) #

The importance of the above theorem is twofold. First, the bound z_{LP1} may be obtained by solving (LP2) and this will be the object of the remaining parts of this section. Secondly, the last inequality relates this bound to classical "assignment bounds" for several applications of the TDTSP model such as the scheduling problem in section 2 and the QAP in [18].

A Dual Problem

Since (LP2) has only n constraints, but an enormous number of variables, it may be solved by column generation. But, for values of $n \geqslant 15$, the convergence of the method becomes too slow to make it practical. Moreover, there is no guarantee that the optimum solution will be integer (see [20] for an

exemple). When using a primal solution method, such as column generation, for solving (LP2) there is usually no good lower bound available before the problem is completely solved. This is an indication that a dual method will be better suited for obtaining good lower bounds for the branch and bound algorithm at reasonable computation expense.

When the shortest path P in N is not a sequence path, we will try to modify the costs on the arcs in the network by adding penalties $\P_i$ in such a way that (i) the length of all the sequence-paths are not changed

and (ii) the shortest path will be induced to visit the jobs previously omitted and to avoid the jobs which were previously included more than once. The penalty $\P_i$ will be incurred by a path each time a node representing job J_i 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:

$$\bar{c}^t_{ij} = c^t_{ij} + \P_i \qquad (1\text{-}5)$$

(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). Rather than actually add on the penalties in this way, it appears to be more efficient to modify the shortest path algorithm to incorporate the penalties: by adding $\P_j$ to f(j,1) and f(j,t) in Step 1 and Step 2a, this is carried out at the expense of only n^2 additionnal computations.

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

$$\sum_{j=1} \pi_j = 0 \qquad (1\text{-}6)$$

Using the vector a^P associated with path P(recall that a_i^P is the number of occurrences of nodes (i,t) representing job J_i in path P), the length of P, in the network with penalties, may be written as

$$\ell_\pi(P) = \ell(P) + \sum_i a_i^P \pi_i \qquad (1\text{-}7)$$

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

$$\ell_\pi(P) = \ell(P) = C_{(w)} \qquad (1\text{-}8)$$

Denoting by C* the cost of an optimal sequence, and by P_π the shorthest path in the network with penalties π, we have:

$$\ell_\pi(P_\pi) \leqslant C^* \qquad (1\text{-}9)$$

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

$$(D) \qquad \text{Max } \{W(\pi) \mid \sum_i \pi_i = 0\}$$

where $W(\pi) = \ell_\pi(P_\pi) = \underset{P}{\text{Min}} \ (\ell(P) + \sum_i a_i^P \pi_i)$.

Theorem 2: Problem (D) is equivalent to the dual of LP2

Proof: (D) may be stated as the linear programming problem:

$$\left[\begin{array}{ll} & \text{Max } W \\ \text{s.t.} & W - \sum_i a_i^P \pi_i \leqslant \ell(P) \qquad \text{all P} \\ & \sum_i \pi_i = 0 \\ & W, \pi_i \text{ unrestricted.} \end{array}\right.$$

The dual of this linear program is:

$$\begin{cases} \text{Min} \sum_P \ell(P)\, x_p \\ \text{s.t.} \quad \sum_P x_p = 1 \\ \quad -\sum_P a_i^p\, x_p + y = 0 \qquad i = 1,2,\ldots,n \\ \quad x \geqslant 0 \end{cases}$$

Since $\sum_i a_i^p = n$ for all P, it follows that $y = 1$ and that the convexity constraint is redundant, so one obtains exactly (LP2) #

Subgradient Optimization:

A subgradient optimization approach (e.g.[6], [14], [15]) to the solution of problem (D) is based on the following observation: for a penalty vector π^k, the vector S^k with components $s_i^k = a_i^p - 1$, where P is a shortest path with respect to π^k, is the projection on the set $\{\pi \mid \sum_i \pi_i = 0\}$ of a subgradient direction for the function W at the point π^k.

Subgradient optimization is now a well-developed technique for obtaining an approximate solution to problems like (D). The application of this technique to (D) is outlined as follows:

Subgradient optimization algorithm:

0 - Select an initial penalty vector π^0; set k = 0.

1 - Compute a shortest path P using π^k.

2 - If P is a sequence-path, then Stop: P is an optimal solution to the TDTSP.

3 - If more iterations are desired, go to step 4, otherwise stop.

4 - Define π^{k+1} using S^k and the previous penalties. Go to step 1.

Different strategies for the stopping criterion used in step 3 and the penalty transformation in step 4 are described in the literature on subgradient optimization. In the experimentation described in Section 2, we used the iteration scheme (see [14]):

$$\pi^0 = 0$$

$$\pi^{k+1} = \pi^k + t_k S^k$$

with

$$t_k = \frac{UB - W(\pi^k)}{||S^k||^2}$$

where UB is an upper bound on the value of an optimal sequence (for instance UB is the cost of a "good" sequence, determined by an heuristic method or another device). This scheme can be seen as an attempt to set the length of the previous solution path to the value UB (if $W(\pi^k) = UB$ for some k, then UB is the value of an optimal solution to the TDTSP).

If termination of the above algorithm occurs in Step 3 with the indication that a shortest sequence path cannot be found at reasonable computation cost with this method, then the information obtained from the solution of shortest paths problems in step 1 will be used within a branch and bound algorithm for solving the TDTSP.

Branch and Bound algorithm:

The set of all n sequence paths in the originalproblem (P) is partitioned

with respect to the last job to be processed. Denote by $k_1, k_2 \ldots, k_s$ the indices of the jobs for which the arcs $((k_r,n), (n+1))$ exist. For the original problem, s=n and $k_r=r$, but implicit bounds and dominance tests, defined below, will allow to reduce the number of these jobs. 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,...,s. The problem (P) is called the "father problem" and the subproblems P_r are called its "sons". The father problem being a TDTSP of size n, the sons are also TDTSP's, of size n-1, and some information (bounds, penalties,...) obtained from the father problem may be used to speed up the work on its sons. For instance, denoting by ¶ the "best" penalties found during the iterations of subgradient optimization in (P), good initial penalties $¶^0$ for P_r may be defined by

$$¶^0_j = ¶_j + \frac{¶_{k_r}}{n-1} \quad \text{for all } j \neq k_r$$

Since branch and bound is a widely used optimization method, the only features in the present algorithm which will be described here are those which are not included in conventional branch and bound algorithms, namely the implicit bounds and the dominance test.

Implicit Bounds:

As a by-product of the algorithm, the 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 ¶. These values B_r are called the implicit bounds for P_r

The implicit bounds allow early fathoming of subproblem 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) is utilized, and makes use of a subsequent ranking of the remaining P_r's, according to non-increasing order of their implicit bounds. This exploration strategy is preferred because of its predictable storage requirement, in contrast with the least bound strategy (jump-tracking).

Another benefit of the implicit bounds is the use of a redirected 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 at the node (k_r,n) with least bound B_r. In this manner, a more specific attempt is made to improve the worst (smallest) implicit bound, and this could be seen as taking advantage of a look-ahead power with respect to the sons of the current problem. The experiments made during the design phases 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 amount of enumeration, a dominance test is introduced for early detection of fathomable 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 is defined by fixing the (n-k) last jobs, say $J_{k+1}, J_{k+2}, \ldots, J_n$. Its sons will be defined by fixing the k^{th} job. Consider J_r as a possible k^{th} 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 an 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 $J_{k+4}, \ldots$, 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}, $\ldots$, J_n is a locally optimal sequence for positions k to n+1 with J_r fixed as the "first" job in this partial sequence.

These tests could be performed before the shortest path iterations in order to reduce the number of possible k^{th} 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, has been convincingly demonstrated in the TSP and the Tardiness Cost problem applications of the model; it resulted in a considerable reduction in the amount of enumeration and computer time required, even 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 [8] 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] for good discussion of this method. It appears that the idea originated in the work of Reiter and Sherman [22] , who described a general heuristic method for discrete optimization. In particular, it provides a good heuristic method for obtaining a good initial solution. It has proven its value for the specific applications to the TSP and the Weighted Tardiness problem.

Discussion:

Computational results for the general TDTSP will not be presented since it was introduced mainly to serve as the model for 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_{ij}^{t} has little meaning, and it appears that specialized problems are much more difficult than random ones.

Indeed, some 10-job problems, with random values for all C_{ij}^{t}, were easily solved usually without enumeration. It should be noted, however, that Fox [10], who used general integer programming, was unable to handle problems of this size.

The next section adresses a general one-machine scheduling problem, for which the TDTSP provesto be a valuable relaxation. Other applicationsof the above methodology will be outlined in the last section.

Section 2. MINIMIZING TARDINESS COSTS IN ONE-MACHINE SCHEDULING

Definitions:

The following one-machine scheduling problem is called the minimum tardiness cost problem (MTCP): n jobs $J_1, J_2, \ldots, J_n$, which arrive simultaneously at time $\tau = 0$, have to be processed on a continuously available machine that can perform only one job at a time.

Associated with each job J_i are:

- the processing time $p_i \geqslant 0$
- a non-decreasing cost function $C_i(\tau)$, which represents the cost incurred if J_i finishes at time τ.

Any processing schedule leads to a finishing time τ_i for each J_i $(i = 1,2,\ldots,n)$; we want to find a sequence $J_{w(1)}, J_{w(2)}, \ldots, J_{w(n)}$ which minimizes the total cost

$$C_{(w)} = C_{w(1)}(\tau_{w(1)}) + \ldots + C_{w(n)}(\tau_{w(n)}).$$

Interpreting d_i as a due date for $J_i (i=1,2,\ldots,n)$ the following cost functions have been considered in the literature:

a) $C_i(\tau) = \begin{cases} 0 & \text{if } \tau \leqslant d_i \\ \alpha_i & \text{otherwise} \end{cases}$

This leads to an NP - complete problem, see [17] . If $\alpha_i = 1$ for all i, then the problem is to minimize the number of late jobs; good solution methods are available for this problem (see [2]).

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

The problem is then 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) $C_i(\tau) = \alpha_i \text{ Max } \{0; \tau - d_i\} + \beta_i \tau$

The problem is then to minimize the sum of the (weighted) tardiness costs and the flow-time costs. If all $\beta_i = 0$, this problem is called the Weighted Tardiness Problem; if in addition all $\alpha_i = 1$, one obtains the Total Tardiness Problem.

The MTCP has received much attention ([2], [4], [5], [11], [23])

In the following analysis the cost functions are allowed to be arbitrary non-decreasing functions, unless otherwise specified.

The Precedence Relation:

Earlier results, originating with the work of Emmons [8], are noteworthy for having defined a precedence relation, that is, a partial order on the job set, such that an optimum sequence exists which satisties this precedence relation. The reader is referred to sections 2-2 and 2-3 of [23] for the definition and determination of such a precedence relation.

This relation will be denoted by R and iRj means that job J_i must be performed before job J_j (not necessarily immediately before) according to R; in this case, job J_j is called a descendant of job J_i and job J_i is called an ascendant of job J_j. A_i and B_i will, respectively, denote the sets of indices of the descendants and the ascendants of job J_i (in the following, A_i and B_i will denote the sets of indices as well as the sets of the corresponding jobs).

Job J_i will be called a predecessor of job J_j if iRj and no job J_k, distinct from J_i and J_j, satisfies iRk and kRj.

Letting S(i,j,t) be the set of sequences which satisfy 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 conditions for $S(i,j,t) \neq \emptyset$. When this set is not empty, a lower bound for tardiness cost incurred by job J_i in any element of S(i,j,t) will be calculated and then used to define a TDTSP relaxation of the MTCP.

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_j must be processed between the t^{th} and $(t+1)^{st}$ position, 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 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:

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

is the exact tardiness cost incurred by job J_j.

If (2-1) holds with 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 $D(i,j) = \{1,2,\ldots,n\} - (B(i,j) \cup A(i,j))$,

we have the following lemma:

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

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

does not hold.

Proof: obvious from the above arguments.

#

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

If (2-2) holds with 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 D(i,j) with least processing time. Since $C_j(\tau)$ is a non decreasing function, we have:

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

A TDTSP Relaxation.

With the lower bound C_{ij}^{t} of Lemma 4, a TDTSP may be used to provide a lower bound on the total tardiness cost. Define a multipartite graph as in Section 1, 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 position nodes (J_i,t) are defined only for those jobs J_i with $B_i = \emptyset$, and $C_{0i}^{0} = C_i(p_i)$; the next position "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.

The last edges $((J_i,n),n+1)$ are dummies and have length equal to 0.

Illustration: The following example is taken from [23] with

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

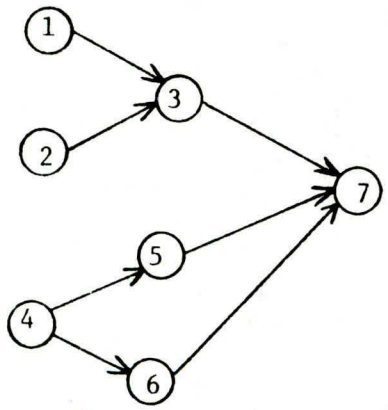
The data are given in Table 2.

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

Data for illustrative MTCP example.

Table 2

The precedence graph is given in Fig.2



Precedence graph for illustrative MTCP example

FIGURE 2

The lower bounds C_{ij}^{t} on the costs and the data for the corresponding multipartite network are given in Table 3. Note that the final node corresponds to job J_7 since, from the precedence graph, J_7 must be processed in the last position.

Position 0

C^0_{ij}	1	2	4
0	0	0	0

Position 3

C^3_{ij}	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

Position 1

C^1_{ij}	1	2	4	5	6
1	x	0	0	x	x
2	0	x	0	x	x
4	0	0	x	0	0

Position 4

C^4_{ij}	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

Position 2

C^2_{ij}	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

Position 5

C^5_{ij}	3	5	6
1	305	x	x
2	305	x	x
3	x	490	264
5	305	x	264
6	305	490	x

Position 6

C^6_{ij}	7
3	0
5	0
6	0

TABLE 3

Cost C^t_{ij} for the illustrative MTCP example.

Theorem 3. 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 completely. A procedure providing a good lower bound on its solution is sufficient and this is performed by the subgradient technique described in Section 1.

Theorem 4. 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,

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

All feasible sequences S (including P*) satisfy $a_i^S = \underline{1}$ for all i. Therefore $\ell(w) \leq \ell(S)$ for all sequences S. From (ii) it follows that $C_{(w)} \leq C_{(S)}$ for all sequences S. #

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

$$\ell(w) \leq C_{(S^*)} \leq C_{(w)}.$$

If condition (i) is not satisfied, a stopping criterion, as defined in Section 1, is used, and the maximum length $\ell(\P^e)$ of a shortest path is then a lower bound on C(S*), i.e.,

$$\ell(\P^e) \leq C_{(S^*)}.$$

Before describing a branch and bound algorithm for the MTCP it is interesting to discuss the value of the lower bound obtained by this approach by comparing it with the lower bound obtained in [23].

Since no attempt is made to solve the TDTSP exactly, the subgradient iterations aim at solving problem (LP2). Here we have a relaxation of the MTCP which is of type 3, the "weakest" form of relaxation. Note that the assignment relaxation described in [23] is of type 1 and may appear tighter.

It is a simple matter to verify that this assignment problem is exactly the relaxation (LAR) of the TDTSP relaxation of the MTCP. Denoting by z_{LP2} the value of an optimum solution to the relaxation (LP2) of MTCP, by z_{LAR} the assignment bound as defined in [23] and by Z* the cost of an optimum solution to the MTCP, it follows from Theorem 1 that

$$z_{LAR} \leq z_{LP2} \leq Z^*$$

Note that the exact value z_{LP2} can be obtained only with optimal penalties, and that subgradient iterations usually stop without providing optimality. But the bound obtained is usually close enough to its optimum value to be likely better than z_{LAR} (it is possible to derive a bound better than z_{LAR} by using penalties derived from the optimal dual variables for LAR, but we did not attempt to use this result). Moreover, the smallest implicit bound is usually a tighter bound than the maximum length of a shortest path, and it is this smallest implicit bound which is actually used for possible fathoming. Additional features for tightening the bounds are described below.

Reductions:

In order to obtain tigher lower bounds for the MTCP, some reductions will be performed in the TDTSP multipartite network associated with it.

The reductions will be performed by deleting one of the two arcs defined by the costs C_{ij}^t and C_{ji}^t, depending on which of two conditions to be described below holds.

Consider J_i and J_j such that no precedence relation holds between them. Denote by θ_{ij}^t 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 the t^{th} and $(t+1)^{th}$ positions. Note that

$$\tau_{ij}^t = \tau_{ji}^t = \theta_{ij}^t + p_i + p_j.$$

Denote 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 5: If, for all $\theta \in [\theta^t_{ij}, \bar{\theta}^t_{ij}]$

we have $$C_j(\theta+p_i\theta+p_j) - C_j(\theta+p_j) \leq C_i(\theta+p_i+p_j) - C_i(\theta+p_i), \quad (2\text{-}3)$$

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 $\theta \in [\theta^t_{ij}, \bar{\theta}^t_{ij}]$ (if t =1, the interval is reduced to the single point 0). Then, the improvement obtained by exchanging J_i and J_j is $(C_i(\theta+p_i) + C_j(\theta+p_i+p_j)) - (C_j(\theta+p_j) + C_i(\theta+p_i+p_j)) \leq 0$. 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 the above relation 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 illustrative example in a following subsection.

The Branch and Bound Algorithm:

The branch and bound algorithm for the MTCP is based on the framework defined in Section 1. The main difference comes from the fact that, for any sequence, its length in the multipartite network is only a lower bound on its cost. When a shortest path happens to be a sequence-path, its length is the best possible bound obtainable in the corresponding subproblem. 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 subproblem, 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 subproblem for further tightening of the bound. This behavior comes from the fact that we have here a relaxation of type 3 for the MTCP, while for the TDTSP the relaxation was of type 2.

The dominance test may also be performed in a more efficient way. Indeed, the cost of assigning job J_r to 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 the first job in the segment.

The Weighted Tardiness Problem:

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

The branch and bound algorithm makes 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 the first operation when defining any subproblem.

The precedence relation is not recomputed, but only copied from the one in the original subproblem. 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 D(i,j) with least processing time. In the same way, the upper bounds $\bar{\theta}_{ij}^t$ are obtained by selecting the jobs in D(i,j) with greatest processing time.

For the reductions, two more precise statements of theorem 5 hold.

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

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

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

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

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

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) $p_i/w_i \leqslant p_j/w_j$

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

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

Proof: Condition (ii) implies

$\theta+ p_i \geqslant d_i$ and $\theta+p_j \geqslant d_j$.

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

Illustration: Corollaries 1 and 2 are applied to the illustrative 7-job example, yielding the reduced network pictured in Fig.3 Note that several arcs and nodes become meaningless: for example, node (2,2) and the arcs issueing from this node may be deleted (no path from the origin to the end may use them). After the deletion of such elements, the network is then reduced to the single sequence-path 2-1-4-5-3-6 with length 454. Clearly the TDTSP problem is then solved in one (trivial) shortest path iteration, and, since condition (ii) of Theorem 4 is satisfied, the MTCP is also solved without branching.

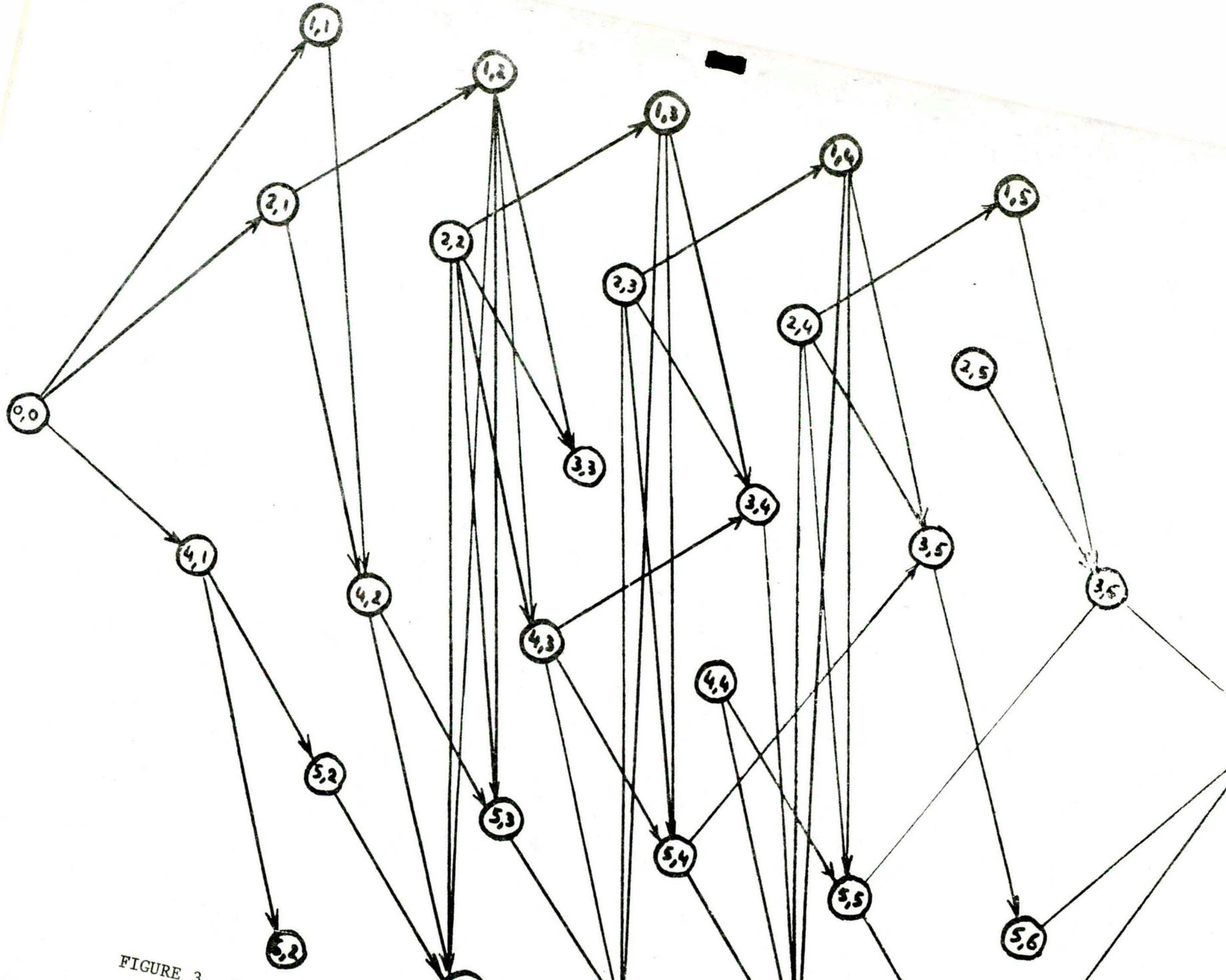


FIGURE 3

Computational Results

The algorithm was coded in FORTRAN for the weighted tardiness problem using a heuristic method based on adjacent pairwise exchanges [2], [22] for computing the initial upper bound UB. It was run on the CDC Cyber 74-18 of the Université de Montréal on 84 test problems, with 15 and 20 jobs. The data for the first set of 42 problems are those of Rinnooy Kan et al. [23], 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 4 and 5.

The mean solution times (respectively, numbers 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 et al., the median and maximum of the solution times and the numbers 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 behavior 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 times and the numbers of nodes are given in table 6.

Considering that the CYBER 74 is reported to be from 2.5 to 3.5 faster than the CYBER 73 used by Rinnooy Kan et al. (hereafter RKLL), the results given in Table 4 show a substantiel 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 times 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 [9] was made. Fisher reports some comparison between his algorithm running on a IMB 360 - 67 and RKLL on a 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.

In a private communication with the authors, Prof. K.R. Baker reports on computational work with L.Schrague, applying their "Chain Algorithm" ([3], [4]) to the same first set of 42 problems, each one being solved within 1.2 seconds on an I.B.M. 370/168. This version of dynamic programming seems to be currently the best way to exploit the precedence relation in the weighted tardiness problem.

TABLE 4. RESULTS FOR THE WEIGHTED TARDINESS PROBLEM

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 5. RESULTS FOR THE WEIGHTED TARDINESS PROBLEM

Number of Nodes in the enumeration tree

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 6. RESULTS FOR THE TOTAL TARDINESS PROBLEM.

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

Possible Improvements:

Some improvements to the existing algorithm are possible. First, attempts to refine the precedence relation could be performed at each node in the enumeration. Second, the arc lengths C_{ij}^{t} in the multipartite network should provide better bounds if the values 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 Thoerem 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 [21], 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 algorithm itself could be improved. Considering that the costs C_{ij}^{n-1} of the last transitions are exact, and that the best permutation 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 positions. The branch and bound algorithm would thus work implicitly two levels down. Of course, this may be extended to triples of jobs in the three last positions, and generally to r-tuples of jobs in the r last positions. It is likely that there is a threshold value for r, after 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 1, the value which is currently used.

3. EXTENSIONS AND CONCLUSIONS

The TDTSP, which was described in Section 1 and applied to the tardiness cost problem in Section 2, may be useful in solving other optimum permutation and related problems.

The first obvious application is to the Traveling Salesman Problem. Actually, this provided the basic impetus for devellopping the general model. The first work in this area was described in two earlier research reports [20],[21] . D. Houck and R. Vemuganti [16] have independently discovered the same approach to the TSP and considered experimental comparisons with the "1-arborescence" approach suggested by Held and Karp [13], for the asymetric problem. 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 [21] for a further description of the routing applications.

Some extensions of the scheduling applications for the one-machine problem were outlined at the end of Section 2. It appears that the method could be extended to problems involving several machines, such as flow-shop and job shop problems, but this will require further examination.

The 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 Section 1. It should be noted that the bounds C_{ij}^t become tighter as the set-up part of the costs become more important. It appears that the method could be extended to problems involving several machines, such as flow-shop and job shop problems, but this will require further examination.

Another optimum permutation problem, somewhat related to one-machine scheduling problems, is the Linear Ordering Problem(see Adolphson and Hu [1]). This problem may be handled as was the weighted tardiness proplem described in Section 2. The main difference lies in the way the bounds defining the multipartite network are computed.

Turning to more general optimum permutation problems, one obtains the Quadratic Assignment Problem (QAP), of which the TDTSP is a special case. The QAP appears to be a very general model as it provides a way to formulate several different optimization problems; it turns out to be a very difficult problem however. At the present time exact solution methods are tractable only for small problems (see [19]). In an application of the TDTSP model to solving the QAP the computation of the bounds used to defined the multipartite network is performed through the solution of related linear assignment problems. These last two applications the (Linear Ordering and Quadratic Assignment Problems) are currently under study.

It appears that much work remains to be done, both on improving the present approach and on 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. They are also indebted to Richard Soland, Alexander Rinnooy Kan and to anonymous referees for their careful reading and their helpful comments on a first version of this paper. 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, "Finding an Optimal Sequence by Dynamic Programming: An Extension to Task Chains," GSBA Paper 145, Graduate School of Business Administration, Duke University, Durham NC (1975).

[4] K.R. BAKER, "Computational Experience with a Sequencing Algorithm Adapted to the Tardiness Problem," GSBA Paper 163, Graduate School of Business Administration, Duke University, Durham NC (1976).

[5] 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).

[6] M.L. BALINSKI and P. WOLFE eds., "Nondiffentiable Optimization," *Math. Programming Study 3* (1975).

[7] R.W. CONWAY W.L. MAXWELL and L.W. MILLER *Theory of Scheduling*, Addison-Wesley, Reading, Mass. (1967).

[8] H. EMMONS, "One-Machine Sequencing to Minimize Certain Functions of Job Tardiness," *Operations Research 17*, 701-715, (1969).

[9] M.L. FISHER, "A Dual Algorithm for the One-Machine Scheduling Problem," O.R. Tech.Rep. 243, Cornell University, Ithaca, N.Y. (1974), (to appear in *Math.Programming*).

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

[11] L.GELDERS and P.R. KLEINDORFER "Coordinating Aggregate and Detailed Scheduling Decisions in the One-Machine Job Shop: Part I. Theory," *Operations Research 22*, 46-60 (1974).

[12] A.M. GEOFFRION, "Lagrangian Relaxation for Integer Programming," *Math. Programming Study 2*, 82-114 (1974).

[13] M. HELD and R.M. KARP, "The Traveling-Salesman Problem and Minimum Spanning Trees," *Operations Research* 18, 1138-1162 (1970).

[14] M. HELD and R.M. KARP, "The Traveling-Salesman Problem and Minimum Spanning Trees," Part II, *Math.Programming 1*, 6-25 (1971).

[15] M. HELD, P. WOLFE and H. CROWDER, "Validation of Subgradient Optimization," *Math. Programming 6*, 62-88 (1974).

[16] D.J. HOUCK and R.R. VEMUGANTI, "The Traveling Salesman Problem and Shortest n-Paths," presented at ORSA-TIMS Meeting, Philadelphia, Pa. May (1976).

[17] 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).

[18] E.L. LAWLER, "The Quadratic Assignment Problem," *Management Science* 19, 586-599 (1963).

[19] 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).

[20] 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).

[21] J.C. PICARD and M. QUEYRANNE "Le problème du Voyageur de Commerce: Plus-Courts Chemins et Optimisation par Sous-Gradients," Rapport Technique EP76-7, Ecole Polytechnique, Montréal (1976).

[22] S. REITER and G. SHERMAN "Discrete Optimizing," *SIAM J.*13, 864-889 (1965).

[23] A.H.G. RINNOOY KAN B.J. LAGEWEG and J.K. LENSTRA, "Minizing Total Costs in One-Machine Scheduling," *Operations Research 23*, 908-927 (1975).

[24] B.ROY, *Algèbre Moderne et Théorie des Graphes*, Dunod, Paris (1970).

A CONSULTER
SUR PLACE