TRABAJOS DE INVESTIGACION OPERATIVA
Vol. 3. Núm. 1, 1988 pp. 117 a 140

ON IMPROVING A SOLUTION TO THE ATSP WITH FIXED ORIGIN AND PRECEDENCE RELATIONSHIPS

L. F. Escudero
IBM GMTC, Sindelfingen, F. R. Germany
and Centro de Invest. UAM-IBM (Madrid)

ABSTRACT

Given the directed graph $G_1 = (N, A_1)$ with a node origin and a penalty matrix C, the ATSP with fixed origin and precedence relationships (hereafter, ATSP-PR) consists of finding the permutation of the nodes from the set N, such that it minimizes a matrix C based function and does not violate the precedence relationships given by the set A_1. In this work we present an algorithm for improving a given feasible solution to the problem, by performing a local search that uses 3- and 4-change based procedures. Computational results on a broad set of cases is reported.

Key words: ATSP, precedence relationships, local search, TRIA and QUAD changes.

Classification: AMS(MOS) subject classification scheme (1970): 90c10, 90c35.

1. PROBLEM DEFINITION AND MOTIVATION

Let the graph $G_1 = (N, A_1)$, where N gives the set of nodes, A_1 gives the set of directed arcs, such that the node $n1$ must be precedent to the node $n2$ for $(n1, n2) \in A_1$ in any feasible solution to the ATSP-PR to be described below and, without loss of generality, there is a node so-called *root*, say 0 such that $\exists n2 \in N | (0, n2) \in A_1$, $\nexists n1 \in N | (n1, 0) \in A_1$, and there is not any other node, say $n2$ from N such that $\nexists n1 \in N | (n1, n2) \in A_1$. The

Recibido: Abril 1987.

ATSP-PR consists of finding a feasible permutation (hereafter, FP) of the nodes such that certain objective function is minimized. Note that a permutation (say, π) is a feasible one if it does not violate the precedence relationships given by the set A_1.

Let $n1 \rightarrow n2$ denote a partial ordering in a given π, such that $\pi_h = n1$ and $\pi_{h+1} = n2$ for $h = 1, 2, ..., |N| - 1$, where π_h denotes the node that belongs to the permutation level (hereafter, *level*) h in π. Let C denote a matrix such that $c_{n1,n2}$ gives the penalty associated with the partial ordering $n1 \rightarrow n2$.

Let the binary matrix S be such that, initially, $s_{n1,n2} = 1$ if, based on the «local» information given by the set A_1 and the matrix C, it has not yet been detected the infeasibility of $n1 \rightarrow n2$; otherwise, $s_{n1,n2} = 0$ for $n1, n2 \in N$. Let $N2(n1)$ (res. $N1(n2)$) define the set of nodes for which it has not yet been detected that they cannot be immediate successors (res. predecessors) of the node $n1$ (res. $n2$) in any FP.

$$N2(n1) = \{n2 \in N | s_{n1,n2} = 1\} \quad \forall n1 \in N \tag{1.1a}$$

$$N1(n2) = \{n1 \in N | s_{n1,n2} = 1\} \quad \forall n2 \in N \tag{1.1.b}$$

The penalty, say $z(\pi)$ associated with a given π can be expressed as follows

$$z(\pi) = \sum_{n1 \in N} \sum_{n2 \in N2(n1)} c_{n1,n2} y_{n1,n2} \tag{1.2}$$

where $y_{n1,n2}$ is a 0-1 variable such that $y_{n1,n2} = 1$ if $n1 \rightarrow n2$ and, otherwise, it is zero. Then, the ATSP-PR consists of

$$\min \{z(\pi) | \pi \text{ is a FP}\} \tag{1.3}$$

The problem has a broad application field; typical applications are as follows:

1. The asymmetric Traveling Salesman problem (see [1, 2, 8] among many others), where every city must be covered just only once, the city-origin is fixed and the visiting of the cities must satisfy some precedence relationships.
2. Flexible Manufacturing Systems (see [3, 7]), where a part type is defined as a list of operations to be performed on identical parts.

Each operation must be performed once and, since there is some flexibility in the execution sequence of the operations, there are precedence relationships. The goal consists of obtaining the sequence in the execution of the operations such that it does not violate the precedence relationships-based constraints, and the set-up/transport cost/time of the complete set of partial orderings $\{n1 \rightarrow n2\}$ is minimized. The penalty $c_{n1,n2}$ can be a raw input of the problem or, more frequently, it can be the result of e.g., a transportation problem. Assume that the operations $n1$ and $n2$ are to be performed in different modules (or working stations), the production proportions are known, and the time/-cost for sending a part from one module to another is also known; then, the problem consists of obtaining the related feasible routing proportions such that the total transport time/-cost $c_{n1,n2}$ is minimized; this penalty is used as the coefficient of the potential partial ordering $n1 \rightarrow n2$ in the objective function.

The paper in organized as follows. Section 2 gives a mathematical formulation of ATSP-PR. Section 3 describes the algorithm that is proposed for improving a FP. And, finally, Section 4 gives some computational experience and conclusions.

2. ATSP-PR's FORMULATION

There is not a unique formulation of (1.2)-(1.3); we give in this section the formulation that seems to be most attractive. Let us introduce some notation that also will be used for other purposes.

Let the binary matrix P have the following meaning: $p_{n1,n2} = 1$ if there is, at least, one predecessor path from the node $n2$ to the node $n1$ in the graph G_1, being such a path $n2$, $pr(n2)$, $pr(pr(n2))$, ..., $pr(.) = n1$ where, say $k = pr(n)$ iff $(k, n) \in A_1$; otherwise, $p_{n1,n2} = 0$.

The matrix S has been defined in Section 1; the element $s_{n1,n2}$ will have, initially, the value 1 if the conditions (2.1)-(2.2) are satisfied by the potential partial ordering $n1 \rightarrow n2$; otherwise, $s_{n1,n2} = 0$ (i.e., $n1 \rightarrow n2$ is not allowed in any FP).

$$c_{n1,n2} \neq \infty \tag{2.1}$$

$$(n1, n2) \in A_1 \vee (p_{n_1,n_2} = 0 \wedge p_{n_2,n_1} = 0) \tag{2.2}$$

Let the augmented graph $G = (N, A)$ be such that $A = A_1 \cup A_2$, where $A_2 = \{n1, 0\}$ such that $n1 \in N$ and $\nexists n2 \in N | (n1, n2) \in A_1$; otherwise, $A = A_1$. Let $s_{n1,0} = 1$ and $c_{n1,0} = 0$ $\forall n1 \in N | (n1, 0) \in A_2$.

Let $h1_n$ and $h2_n$ give the lowest and highest levels for the node n in any FP, respectively. Then,

$$h1_n = 1 + |H1_n| \tag{2.3a}$$

$$h2_n = |N| - |H2_n| \tag{2.3b}$$

where the sets $H1_n$ and $H2_n$ are defined as follows.

$$H1_n = \{n1 \in N | p_{n1,n} = 1\} \tag{2.4a}$$

$$H2_n = \{n2 \in N | p_{n,n2} = 1\} \tag{2.4b}$$

Let $x_{n,h}$ be a 0-1 variable such that $x_{n,h} = 1$ if $\pi_h = n$ and, otherwise, it is zero; let $y_{n1,n2}$ be also a binary variable with the same meaning as in (1.2). Then, «a priori» range of the variables will be

$$x_{n,h} \in \{0, 1\} \text{ for } h = h1_n, ..., h2_n \text{ and, otherwise, } x_{n,h} = 0 \tag{2.5}$$

$$y_{n1,n2} \in \{0, 1\} \quad \forall n1, n2 \in N | s_{n1,n2} = 1 \text{ and, otherwise, } y_{n1,n2} = 0 \tag{2.6}$$

Note that $s_{n1,n2} = 1$ means that there is a level h such that $h1_{n1} \leqslant h \leqslant h2_{n1}$ and $h1_{n2} \leqslant h + 1 \leqslant h2_{n2}$. The ATSP-PR can be formulated as follows

$$\text{mín } z = \sum_{n1,n2 \in N} c_{n1,n2} y_{n1,n2} \tag{2.7}$$

subject to (2.5)-(2.6), and

$$\sum_{n2 \in N2(n2)} y_{n1,n2} = 1 \quad \forall n1 \in N \tag{2.8}$$

$$\sum_{n1 \in N1(n2)} y_{n_1,n_2} = 1 \quad \forall n2 \in N \tag{2.9}$$

$$\sum_{n1,n2 \in T} y_{n1,n2} \leqslant |T| - 1 \quad T \subset N, \quad 2 < |T| < |N| \tag{2.10}$$

$$\sum_{h \in N} x_{n,h} = 1 \quad \forall n \in N \tag{2.11}$$

$$\sum_{n \in N} x_{n,h} = 1 \quad \forall h \in N \tag{2.12}$$

$$\sum_{t1 \in N} t1 \times x_{n1,t1} + 1 \leqslant \sum_{t2 \in N} t2 \times x_{n2,t2} \quad \forall (n1, n2) \in A_1 \tag{2.13}$$

$$x_{n1,h} + x_{n2,h+1} \leqslant 1 + y_{n1,n2} \ \forall n1, n2 \in N | s_{n1,n2} = 1, \quad h = 1, 2, ..., |N| - 1 \tag{2.14}$$

$$x_{n1,h} + x_{n2,h+1} \leqslant 1 \ \forall n1, n2 \in N | s_{n1,n2} = 0, \ h = 1, 2, ..., |N| - 1 \tag{2.15}$$

Two blocks can be recognized in the above model. The constraints (2.8)-(2.10) are related to the ATSP, and (2.11)-(2.16) are related to the precedence relationships. (2.8) (res. (2.9)) avoid more than one partial ordering $n1 \rightarrow n2$ for any $n1$ (res. $n2$), and (2.10) avoid subtours. (2.11) (res. (2.12)) avoid more than one level (res. node) for a given node (i.e., level). (2.13) prevent that a node have a lower level than the level of any node that must be precedent. (2.14) force the penalty $c_{n1,n2}$ if $n1 \rightarrow n2$ belong to the solution. (2.15) avoid the partial ordering $n1 \rightarrow n2$ if it is not allowed in any FP.

Solving the above problem may require more CPU time than allowed. If the optimality is required the best approach seems to be using a cutting planes based LP relaxation for obtaining a tight lower bound on the optimal solution and fixing variables and next, if required, using the branch-and-bound methodology for obtaining the optimal solution; see in [4, 5] the framework for the LP-based combinatorial problem solving.

The proposed (inexact) algorithm for solving ATSP-PR (1.2)-(1.3) is as follows:

1. Obtaining the matrix P and initial sets $N2(n1)$ and $N1(n2)$ $\forall n1$, $n2 \in N$ by using the information provided by the set A and the matrix C.
2. Preprocessing. A strong characterization of the potential partial orderings $\{n1 \rightarrow n2\}$ should be performed, such that the feasibility of any $n1 \rightarrow n2$ is analyzed; i.e., the goal consists of reducing

as much as possible the cardinality of the sets $N2(n1)$ and $N1(n2)$.

3. Obtaining an initial FP. It is given by the vector $\{x(n1)\ \forall n1 \in N\}$, where $x(n1)$ gives the node, say $n2$ if $\pi_h = n1$ and $\pi_{h+1} = n2$ for any h, such that $h = 1, 2, ..., |N|$ and assuming that $h + 1 = 1$ *for* $h = |N|$. Let $y(n2) = n1$ iff $x(n1) = n2$. Let $\bar{z}$ denote the value of FP in the objective function.
4. Improving the FP. See Section 3.

3. IMPROVING THE FP

This section describes the procedures for improving the initial FP; basically, they consist of performing a local search for obtaining a hamiltonian circuit, say $\mathscr{H}$ with a better objective function value while preserving the precedence relations given by the matrix P and the constraints given by the matrix S. The main ideas for selecting the partial orderings to be substituted are described in [6] for the ATSP. We use the procedures so-called TRIA and QUAD; TRIA works on the nodes, say $n1$, $n2$ and $n3$ such that the new partial orderings are (see Figure 1) $x(n1) = n2$, $x(yn2) = n3$ and $x(\text{yn3}) = xn1$, where ya and sb give the current nodes $y(a)$ and $x(b)$, respectively; QUAD works on the nodes, say $n1$, $n2$, $n3$ and $n4$ such that the new partial orderings are (see Figure 2) $x(n1) = n2$, $x(yn2) = xn1$, $x(n3) = n4$ and $x(yn4) = xn3$. Contrary to the algorithm described in [6], each TRIA iteration does not perform more than three changes for obtaining a new $\mathscr{H}$; analyzing the feasibility of a given π is time consuming and trying more than three changes could be prohibitive for problems with size, say $|N| > 50$. Based on our computational results, we recommend to use TRIA first and, when it can not improve the solution, try QUAD and repeat the combination till no better solution can be found.

See in [9] a procedure for locally improving a solution to the Dial-A-Ride Problem (hereafter, DARP); the procedure performs k-interchanges for $k = 2$ and $k = 3$ (TRIA). Basically, DARP consists of the Symmetric TSP (hereafter, STSP) where $G_1 = (N, A_1)$ is a bipartite graph being $N = N_1 \cup N_2$, where any node from the set N_2 has only one predecessor (and it must be from the set N_1) and any node from the set N_1 has only one successor (and it must be from the set N_2). The

algorithm makes use of the special structure of the very specific precedence relationships of the problem. Its complexity is $O(|N|^k)$; the same complexity as the k-interchange procedure for the STSP without precedence relationships. The ATSP-PR allows a general graph G_1 and has a complexity of $O(|N|^4)$; however, given the cuts performed in the TRIA and QUAD procedures (see below), the expected number of iterations is not very high and, in any case, the algorithm likely provides acceptable solutions with an affordable computational effort for problems up to 100 nodes (see Section 4).

3.1. THE TRIA PROCEDURE

The partial ordering $n1 \rightarrow n2$ is named an *admissible* one if, while choosing $n2$ for a given $n1$, there is not evidence that $n1 \rightarrow n2$ cannot be selected as the locally most favorable substitution for the partial ordering $n1 \rightarrow x(n1)$. Formally, $n1 \rightarrow n2$ is an admissible partial ordering if condition (3.1) is satisfied.

$$s_{n1,n2} = 1 \wedge n2 \neq x(n1) \wedge c_{n1,x(n1)} > c_{n1,n2} \tag{3.1}$$

Note that the admissible partial ordering, say $n1 \rightarrow n2$ «cuts» the *incumbent* $\mathcal{H}$, so that the cycle $n1 \rightarrow n2 \rightarrow x(n2) \rightarrow \cdots \rightarrow y(n1) \rightarrow n1$ is introduced. Let $n3$ denote the incumbent node that breaks this cycle and, then, creates a new $\mathcal{H}$ (see Figure 1) that additionally improves the objective function value. Formally,

$$\begin{aligned} n3 = \arg\max \{ & c_{n1,x(n1)} - c_{n1,n2} + c_{y(n2),n2} - c_{y(n2),n3c} + \\ & + c_{y(n3c),n3c} - c_{y(n3c),x(n1)} > 0\} \end{aligned} \tag{3.2}$$

subject to (3.3) and (3.4).

$$n3c \in sp(x(n2) \rightarrow n1) | s_{y(n2),n3c} = 1 \wedge s_{y(n3c),x(n1)} = 1 \tag{3.3}$$

$$\text{The new } \mathcal{H} \text{ is a FP} \tag{3.4}$$

where $sp(a \rightarrow b)$ denotes the set of nodes in the *successor path* $a \rightarrow x(a) \cdots \rightarrow y(b) \rightarrow (b)$.

Let g' denote the reduction on the objective function value due to the $n1$, $n2$, $n3$-based TRIA change; it is given by the expression that is maximized in (3.2).

Recall that 0 is the unique *root* node in the graph G_1; see Section 1. The new $\mathscr{H}$ is *not* a FP if the following conditions are satisfied:

Case 1: $0 \in sp(x(n1) \rightarrow y(n2))$.

$$\exists n \in sp(n2 \rightarrow y(n3c)) \wedge \exists nn \in sp(n3c \rightarrow n1) | p_{n,nn} = 1 \tag{3.5}$$

Note that any node from $sp(n3c \rightarrow n1)$ has a lower level than any node from $sp(n2 \rightarrow y(n3c))$ in the new $\mathscr{H}$. In this case, it is said that $sp(n3c \rightarrow n1)$ *has a lower level than* $sp(n2 \rightarrow y(n3c))$.

Case 2: $0 \in sp(n2 \rightarrow y(n3c))$.

$$\exists n \in sp(n3c \rightarrow n1) \wedge \exists nn \in sp(x(n1) \rightarrow y(n2)) | p_{n,nn} = 1 \tag{3.6}$$

Case 3: $0 \in sp(n3c \rightarrow n1)$.

$$\exists n \in sp(x(n1) \rightarrow y(n2)) \wedge \exists nn \in sp(n2 \rightarrow y(n3c)) | p_{n,nn} = 1 \tag{3.7}$$

The order on which the *candidate* node $n3c$ is chosen from $sp(x(n2) \rightarrow \rightarrow n1)$, and the nodes n and nn are chosen from $sp(n2 \rightarrow n1)$, so that the conditions (3.5)-(3.7) are tested, has a strong influence in the computing time required by the TRIA procedure outlined above. For the case 1 we use the order $n1, y(n1), \ldots$ for $n3c$ and nn, and the order $n2, x(n2), \ldots$ for n; for the case 2 the order is $n1, y(n1), \ldots$ for $n3c$ and n; and for the case 3 the order is $x(n2)$, $x(x(n2)), \ldots$ for $n3c$ and nn. An appropriate order is necessary but not sufficient for avoiding unnecessary operations in the feasibility testing of the new $\mathscr{H}$. We can see that e.g. the testing of the condition (3.5) with the node $nn = n1$ is repeated for every candidate node $n3c$ given $n1 \rightarrow n2$. The following procedure avoids the repetition: (1) Preserve the above given order for selecting the nodes; (2) Let the node f be such that $x(f)$ is the last node nn that was used while testing (3.5) for the previous candidate node; (3) if $x(f)$ *did not* satisfy (3.5) (what means that $x(f)$ is the previous candidate node) then let $n3c$ be the same

node f and, otherwise, $n3c$ will be the node n for which $x(f)$ *did* satisfy (3.5); (4) Restrict the set $\{nn\}$ to $sp(n3c \rightarrow f)$ while testing (3.5) for $n3c$. Similar approaches are taken for the cases 2 and 3. Since the conditions (3.5)-(3.7) are tested for every candidate node, a mechanism similar to the one described above must be included in any competitive implementation of TRIA.

The complexity of TRIA is $O(|N^4|)$, since the whole process is dominated by Step (3) (see below). The number of major iterations in this step is smaller than $|N^2|$; each iteration is devoted to a pair $(n1, n2)$ of nodes. The routine «Selecting the node $n3$ related to the nodes $n1$ and $n2$» has a complexity of $O(|N^2|)$ and dominates its related major iteration; in any case, based on the reasons given above, the expected of iterations is smaller than $|N^2|$.

The algorithm is as follows.

Step (1): Classifying the potential partial orderings where $n1$ is the starting node $\forall n1 \in N$, according to the criterion of non-decreasing cost. I.e., let the vector $IY(n1,.)$ give the ordered set of nodes $\{n2 \in N | s_{n1,n2} = 1\}$ such that $c_{n1,n2} \leqslant c_{n1,n3}$ for $iy(n1,i) = n2, iy(n1,j) = n3$, $i < j$ and $i,j = 1, 2, ..., |N(n1)|$. Note that $N2(n1)$ is given by (1.1).

Step (2): Classifying the partial orderings in the incumbent FP, according to the criterion of non-increasing cost. I.e., let the vector IX give the ordered set of nodes from N such that $c_{n1,x(n1)} \geqslant c_{n2,x(n2)}$ for $ix(i) = n1$, $ix(j) = n2$, $i < j$ and $i,j = 1, 2, ..., |N|$.

Step (3): For each $i1 = 1, 2, ..., |N|$ and $i2 = 1, 2, ..., |N2(ix(i1))|$:

Selecting the nodes $n1$ *and* $n2$.

$n1 := ix(i1)$, $n2 := iy(n1, i2)$

if $s_{n1,n2} = 0 \vee n2 = x(n1) \vee c_{n1,x(n1)} \leqslant c_{n1,n2}$ then go to e3

/*select a new pair*/

$n3 := -1$, $g' := 0$, $g := c_{n1,x(n1)} - c_{n1,n2} + c_{y(n2),n2}$

if $n1 = 0$ then: $n3c := x(n2)$, $f := n2$, case: = 3

else: $n3c := n1$, $f := n1$

if $0 \in sp(x(n1) \rightarrow y(n2))$ then *case*: = 1, else *case*: = 2

Selecting the node n3 related to the nodes n1 and n2. Based on reasons given above, we begin analyzing $n1$, $y(n1)$, ... *for case* $\neq 3$ and $x(n2)$, $x(x(n2))$, ... for *case* $= 3$ as the candidate node $n3c$.

p1: if $s_{y(n2),n3c} = 0 \wedge s_{y(n3c),x(n1)} = 0$ then go to ep1

$a = g - c_{y(n2),n3c} + c_{y(n3c),n3c} - c_{y(n3c),x(n1)}$

if $a \leqslant g'$ then go to ep1 /*select a new $n3c$*/

if *case* $= 1$ then: /*analyze if $sp(n3c \rightarrow n1)$ may have a lower level than $sp(n2 \rightarrow y(n3c))$*/

$nn: = f$

s1: $n: = n2$

if $p_{n,nn} = 1$ then go to p2, else go to s12

s11: if $p_{n,nn} = 1$ then $f: = y(nn)$, $n3c: = n$, go to p1

s12: if $n \neq y(n3c)$ then: $n: = x(n)$, go to s11

if $nn \neq n3c$ then: $nn: = y(nn)$, go to s1 /*select a new $n3c$*/

$f: = y(n3c)$

if *case* $= 2$ then: /*analyze if $sp(x(n1) \rightarrow y(n2))$ may have a lower level than $sp(n3 \rightarrow n1)$*/

$n: = f$

s2: if $\exists nn \in sp(x(n1) \rightarrow y(n2)) | p_{n,nn} = 1$ then: $n3c: = x(0)$, *go to ep1*

if $n \neq n3c$ then: $n: = y(n)$, go to s2

f: $= y(n3c)$

if *case* $= 3$ then:
/*analyze if $sp(n2 \rightarrow y(n3c))$ may have a lower level than $sp(x(n1) \rightarrow y(n2))$*/

$nn: = f$

s3: if $\exists n \in sp(x(n1) \rightarrow y(n2)) | p_{n,nn} = 1$ then go to p2

if $nn \neq y(n3c)$ then: $nn: = x(nn)$, go to s3

$f: = n3c$

$n3 := n3c$, $g' := a$ /*$n1, n2, n3$-based incumbent TRIA change*/
ep1: if $case = 1$ then:

if $n3c \neq x(n2)$ then: $n3c := y(n3c)$, go to p1

if $case = 2$ then:

if $n3c \neq x(0)$ then: $n3c := y(n3c)$, go to p1

if $n2 \neq 0$ then: $n3c := x(n2)$, $f := n2$, $case := 3$, go to p1

if $case = 3$ then:

if $n3c \neq 0$ then: $n3c := x(n3c)$, go to p1

(The step p1 to ep1 analyzes if the proposed $n1$, $n2$, $n3c$-based TRIA change satisfies any of the constraints (3.5)-(3.7); if so, it is rejected. Note that the f-based mechanism and, then, the order on which the nodes in $sp(n2 \rightarrow n1)$ are analyzed prevents the repetition of the feasibility testing for the nodes that belong to this path in the current FP; it strongly reduces the computational time).

p2: if $n3 = -1$ then go to Step (4)

e3: *end* of the Step (3) for the current $i1$ and $i2$. If all $i2$ and $i1$ have been analyzed, stop.

Step (4): *A new FP has been found*

$\bar{z} := \bar{z} - g'$

Performing the TRIA change for obtaining the new FP:

$xn1 := x(n1)$, $yn2 := y(n2)$, $yn3 := y(n3)$

$x(n1) := n2$, $y(n2) := n1$, $x(yn2) := n3$, $y(n3) := yn2$,
$x(yn3) := xn1$, $y(xn1) := yn3$

Go to Step (2)

3.2. THE QUAD PROCEDURE

The notation used by QUAD is consistent with the one introduced in Section 3.1. Let $y(n2) \rightarrow x(n1)$ denote the partial ordering implied by

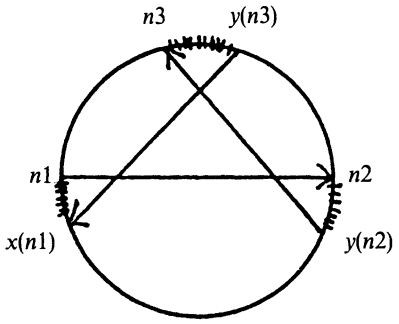

Figure 1. TRIA change

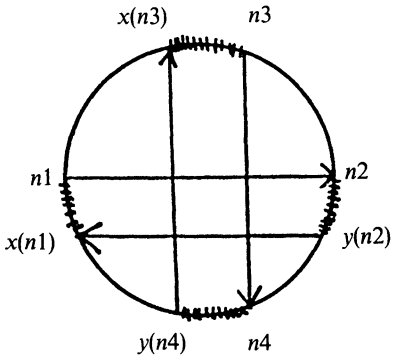

Figure 2. QUAD change

$n1 \rightarrow n2$ in any QUAD change. Note that the new substitutions produce two cycles, namely $n1 \rightarrow n2 \rightarrow x(n2) \rightarrow \cdots \rightarrow y(n1) \rightarrow n1$ and $x(n1) \rightarrow x(x(n1)) \rightarrow \cdots \rightarrow y(y(n2)) \rightarrow y(n2) \rightarrow x(n1)$. Let the following additional notation: The matrix $G12$ will be such that its element $g12_{a,b}$ for $a, b \in N$ gives the «gain» obtained by introducing the partial ordering $a \rightarrow b$, such that

$$g_{a,b} = c_{a,x(a)} - c_{a,b} + c_{y(b),b} - c_{y(b),x(a)} \tag{3.8}$$

The potential partial ordering $n1 \rightarrow n2$ is termed *admissible* if condition (3.9) is satisfied.

$$s_{n1,n2} = 1 \vee s_{y(n2),x(n1)} = 1 \wedge n2 \neq x(n1) \wedge n2 \neq x(x(n1)) \wedge \\ \wedge g12_{n1,n2} > 0 \tag{3.9}$$

It is assumed that the QUAD change based on $n1 \rightarrow n2 | g12_{n1,n2} \leqslant 0$ cannot likely produce a better FP; if $g12_{n1,n2} > 0$ we have, at least, some expectation of improving the solution.

Let $n3$ and $n4$ denote the nodes, such that the partial orderings $n3 \rightarrow n4$ and $y(n4) \rightarrow x(n3)$ produce a new $\mathscr{H}$, by «cutting» the two cycles produced by $n1 \rightarrow n2$ (see Figure 2), that additionally improves the objective function value. Formally, $n3 \rightarrow n4$ must satisfy conditions (3.10)-(3.13).

$$n3 \in sp(n2 \rightarrow y(n1)) \tag{3.10}$$

$$n4 \in sp(x(x(n1)) \rightarrow y(n2)) \tag{3.11}$$

(Note that even for the $\mathscr{H}: a \rightarrow x(a) \rightarrow y(b) \rightarrow b \rightarrow a$, a QUAD change is allowed; namely, $n1 = a$, $n2 = b$, $n3 = b$, $n4 = y(b)$ such that the new $\mathscr{H}$ will be $a \rightarrow b \rightarrow y(b) \rightarrow x(a) \rightarrow a$).

$$s_{n3,n4} = 1 \wedge s_{y(n4),x(3)} = 1 \wedge g12_{n1,n2} + g12_{n3,n4} > 0 \tag{3.12}$$

The new $\mathscr{H}$ is a FP (3.13)

The new $\mathscr{H}$ is *not* a FP if the following conditions are satisfied.

Case 1: $0 \in sp(x(n3) \rightarrow n1)$.

$$\text{Either } \exists n \in sp(x(n1) \rightarrow y(n2)) \wedge \exists nn \in sp(n2 \rightarrow n3) | p_{n,nn} = 1 \tag{3.14}$$

$$\text{or } \exists n \in sp(x(n1) \rightarrow y(n4)) \wedge \exists nn \in sp(n4 \rightarrow y(n2)) | p_{n,nn} = 1 \tag{3.15}$$

Note that $sp(n2 \rightarrow n3)$ has a lower level than $sp(n4 \rightarrow y(n2))$ in the new $\mathscr{H}$, and the level of $sp(n4 \rightarrow y(n2)$ will be lower than the level of $sp(x(n1) \rightarrow y(n4))$.

Case 2: $0 \in sp(x(n1) \rightarrow y(n4))$.

$$\text{Either } \exists n \in sp(n4 \rightarrow y(n2)) \wedge \exists nn \in sp(n2 \rightarrow n1) | p_{n,nn} = 1 \tag{3.14}$$

$$\text{or } \exists n \in sp(n2 \rightarrow n3) \wedge \exists nn \in sp(x(n3) \rightarrow n1) | p_{n,nn} = 1 \tag{3.17}$$

Case 3: $0 \in sp(n4 \rightarrow y(n2))$.

Either $\exists n \in sp(n2 \rightarrow n1) \wedge \exists nn \in sp(x(n1) \rightarrow y(n4)) | p_{n,nn} = 1$ (3.18)

or condition (3.17) is satisfied

Case 4: $0 \in sp(n2 \rightarrow n3)$.

Either $\exists n \in sp(x(n3) \rightarrow n1) \wedge \exists nn \in sp(x(n1) \rightarrow y(n2)) | p_{n,nn} = 1$ (3)19)

or condition (3.15) is satisfied

An efficient implementation of the QUAD procedure given by (3.8)-(3.19) must include a mechanism very similar to the one described in Section 3.1.

The complexity of QUAD is $O(|N^4|)$; the process is dominated by Step (2) (see below). The number of major iterations in this step is smaller than $|N^2|$; each iteration is devoted to a pair $(n1, n2)$ of nodes. The routines *qfeasn*4 or, alternatively, *qfeasn*3 are executed at each major iteration; based on the same reasons given for the TRIA procedure, the expected number of iterations is smaller than its maximum $|N^2|$.

The algorithm is as follows.

Step (1): Obtaining the «gain» matrix G12 (3.8).

Step (2): Obtaining a QUAD change:

$n1 := 0$ /*beginning with the node 0*/

r: if $|N2(n1)| = 1$ then go to er /*try a new n1*/

if $x(n1) = 0 \wedge x(x(n1)) = 0$

then $ind14 := 0$ /*cases 2 and 3*/

else $ind14 := 1$ /*cases 1 and 4*/

$n2 := x(x(x(n1)))$

r0: if condition (3.9) is *not* satisfied then:

if $n2 = 0$ then $ind14 := 0$

go to er0 /*try a new n2*/

if $ind14 = 0$ then go to c32

c14: /*cases 1 and 4*/

call qfeasn4 /*analyzing condition (3.15). It also obtains the vector FN*/

if $feas = 0$ the go to er0 /*no feasible node n4*/

case 1:

$n3 := n2, f := n3$

d1: if $|N2(n3)| = 1 \vee |N1(x(n3))| = 1$ then go to ed1

if condition (3.14) is satisfied for $nn \in sp(f \rightarrow n3)$ then go to *case* 4

if $\exists n4 | fn(n4) = 1$ such that conditions (3.11) and (3.12) are satisfied then go to Step (3)

$f := x(n3)$

ed1: if $x(n3) \neq 0$ then: $n3 = x(n3)$, go to d1

case 4:

$n3 := y(n1), f := x(n3)$

d4: = if $|N2(n3)| = 1 \vee |N1(x(n3))| = 1$ then go to ed4

if condition (3.19) is satisfied for $n \in sp(x(n3) \rightarrow f)$ then go to er0

if $\exists n4 | fn(n4) = 1$ such that conditions (3,11) and (3.12) are satisfied then go to Step (3)

$f := n3$

ed4: if $n3 \neq 0$ then: $n3 := y(n3)$, go to d4

go to er0

c32: /*cases 3 and 2*/

call qfeasn3 /*analyzing condition (3.17). It also obtains the vector FN*/

if $feas = 0$ the go to er0 /*no feasible node n3*/

case 3:

$n4: = x(x(n1))$, $f: = y(n4)$

d4: if $|N2(y(n4))| = 1 \vee |N1(n4)| = 1$ then go to ed3

if condition (3.18) is satisfied for $nn \in sp(f \rightarrow y(n4))$ then go to *case* 2

if $\exists n3 | fn(n3) = 1$ such that conditions (3.10) and (3.12) are satisfied then go to Step (3)

$f: = n4$

ed4: if $n4 \neq 0$ then: $n4: = x(n4)$, go to d4

case 2:

$n4: = y(n2)$, $f: = n4$

d2: if $|N2(y(n4))| = 1 \vee |N1(n4)| = 1$ then go to ed2

if condition (3.16) is satisfied for $n \in sp(n4 \rightarrow f)$ the go to er0

if $\exists n3 | fn(n3) = 1$ such that conditions (3.10) and (3.12) are satisfied the go to Step (3)

$f: = y(n4)$

ed2: if $y(n4) \neq 0$ then: $n4 = y(n4)$, go to d2

er0: if $x(n2) \neq n1$ then: $n2: = x(n2)$, go to r0
/*repeat step r0 to er0 for the new n2*/

er: if $x(n1) \neq 0$ then: $n1: = x(n1)$, go to r
/*repeat step r to er for the new n1*/

stop

Step 3: *A new FP has been found.*

$\bar{z}: = \bar{z} - g12_{n1,n2} - g12_{n3,n4}$

Perform the QUAD change; update the matrix $G12$; and go to Step (2) to analyze a new QUAD change.

*qfeasn*4

Cases 1 and 4 produce a non-feasible QUAD change for $n1 \to n2$ if, independently of the node $n3$, the node $n4$ for $n4 \in sp(x(x(n1)) \to y(n2))$ satisfies condition (3.15). Let the vector FN be such that $fn(n) = 0$ means that the testing is positive; otherwise, $fn(n4) = 1$. If $\exists n4$ that does not satisfy the condition then $feas = 1$ (i.e., it is yet possible that $n1 \to n2$ can produce a feasible QUAD change.); otherwise, $feas = 0$. Let use the order $y(n2)$, $y(y(n2))$, ... for selecting the nodes $n4$ and nn, and the order $x(n1)$, $x(x(n1))$, ... for selecting the node n. The procedure for testing condition (3.15) is as follows.

q4: $feas = 0$

$n4 := y(n2)$, $f := n4$

q41: $nn := f$

q42: $n := x(n1)$

q43: if $p_{n,nn} = 1$ then:

fn(nn): $= 0$ $\forall nn \in sp(x(n) \to n4)$

if $n = x(n1)$ then: go to eq4

$f := y(f)$, $n4 := x(n)$, go to q44

if $n \neq y(n4)$ then: $n := x(n)$, go to q43

if $nn \neq n4$ then: $nn := y(nn)$, go to q42

$fn(n4) := 1$, $feas := 1$, $f := y(n4)$

q44: if $y(n4) \neq x(n1)$ then: $n4 = y(n4)$, go to q41

eq4: return

Note that the f- based mechanism avoids the repetition of the testing of condition (3.15) for the nodes in $sp(x(f) \to y(n2))$.

*qfeasn*3

Cases 2 and 3 produce a non-feasible QUAD change for $n1 \to n2$ if, independently of the node n4, the node n3 for $n3 \in sp(n2 \to y(n1))$

satisfies condition (3.17). Let the vector FN be such that $fn(n3) = 0$ means that the testing is positive; otherwise, $fn(n3) = 1$. If $\exists n3$ that does not satisfy the condition then $feas = 1$ li.e., it is yet possible that $n1 \rightarrow n2$ can produce a feasible QUAD change); otherwise, $feas = 0$. Note that the above approach is very similar for *qfeasn*3 and *qfeasn*4. The procedure for testing condition (3.17) is as follows.

q3: $feas = 0$

$n3: = n2$, $f: = n3$

q31: $n: = f$

q32: $nn: = n1$

q33: if $p_{n,nn} = 1$ then:

$fn(n): = 0 \ \forall n \in sp(y(nn) \rightarrow n3)$

if $nn = n1$ then: go to eq3

$f: = x(f)$, $n3: = y(nn)$, go to q34

if $nn \neq x(n3)$ then: $nn: = y(nn)$, go to q33

if $n \neq n3$ then: $n: = x(n)$, go to q32

$fn(n3): = 1$, $feas: = 1$, $f: = x(n3)$

q34: if $x(n3) \neq n1$ then: $n3: = x(n3)$, go to q31

eq3: return

3.3. THE 2-INTERCHANGE PROCEDURE

It consists of substituting the $sp(n1 \rightarrow x(n2))$ of a FP by the new path defined by $n1 \rightarrow n2 \rightarrow y(n2) \rightarrow \cdots \rightarrow x(n1) \rightarrow x(n2)$; that is, it substitutes $n1 \rightarrow x(n1)$ and $n2 \rightarrow x(n2)$ by $n1 \rightarrow n2$ and $x(n1) \rightarrow x(n2)$, respectively and reverse the $sp(x(n1) \rightarrow y(n2))$. The partial ordering $n1 \rightarrow n2$ is an *admissible* one if condition (3.20) is satisfied.

$$s_{n1,n2} = 1 \wedge s_{x(n1),x(n2)} = 1 \qquad (3.20a)$$

$$c_{n1,x(n1)} + c_{n2,x(n2)} - c_{n1,n2} - c_{x(n1),x(n2)} > 0 \qquad (3.20b)$$

The new $\mathscr{H}$ is *not* a FP if the following conditions are satisfied.

Case 1: $0 \in sp(x(n2) \to n1)$.

$$\exists n \in sp(x(n1) \to y(n2)) \wedge \exists nn \in sp(x(n) \to n2) | p_{n,nn} = 1 \qquad (3.21)$$

Case 2: $0 \in sp(x(n1) \to n2)$.

$$\text{Either } \exists n \in sp(x(0) \to n2) \wedge \exists nn \in sp(x(n) \to y(0)) | p_{n,nn} = 1 \qquad (3.22)$$

$$\text{or } \exists n \in sp(x(n2) \to n1) \wedge \exists nn \in sp(x(n1) \to y(0)) | p_{n,nn} = 1 \qquad (3.23)$$

$$\text{or } \exists n \in sp(x(n1) \to y(y(0))) \wedge \exists nn \in sp(x(n) \to y(0)) | p_{n,nn} = 1 \qquad (3.24)$$

The new $\mathscr{H}$ will improve the current FP if condition (3.25) is satisfied.

$$\left(\sum_{n \in sp(n1 \to n2)} c_{n,x(n)} \right) - c_{n1,n2} - \left(\sum_{n \in sp(x(x(n1)) \to n2)} c_{n,y(n)} \right) - c_{x(n1),x(n2)} > 0 \qquad (3.25)$$

Remarks. The complexity is $O(|N^4|)$, but the expected number of iterations is greater than the expected number for the TRIA and QUAD procedures. The improving testing (3.25) is not cheap. Note that the testing (3.20) is sufficient for the STSP. Only a very limited testing (3.21)-(3.25) is required by the DARP [9] In conclusion, the 2-interchange procedure seems attractive and is frequently used in the STSP, but is not recommended for the ATSP-PR.

4. COMPUTATIONAL EXPERIENCE. CONCLUSION

For illustrative purposes, Figure 3 and Table 1 show a case with $|N| = 8$ nodes; Tables 2 and 3 give the matrices P and S. The FP proposed by the algorithm is as follows: $0 \to 1 \to 4 \to 2 \to 7 \to 6 \to 5 \to 3$ with $\bar{z} = 21.25$; by inspection, we can see that it is an optimal solution to ATSP-PR.

The algorithm described in this paper was tested by implementing a prototype running on the IBM 4381 VM/SP-4; it was written in PL/I and compiled with the version 1.3.

The set of problems whose results are reported consists of nine

classes with sizes $|N| = 8, 32, 37, 59, 69, 72, 85, 92$ and 99; each class is included by one real-life problem and ten generated problems. The latter ones differ from the real-life problem of each class in the penalty coefficients; such that where $c_{n1,n2} \neq \infty$ in the real-life problem, the coefficients for the generated problems were drawn from a uniform distribution with the range $[0{,}9 \times c_{n1,n2},\ 1.1 \times c_{n1,n2}]$.

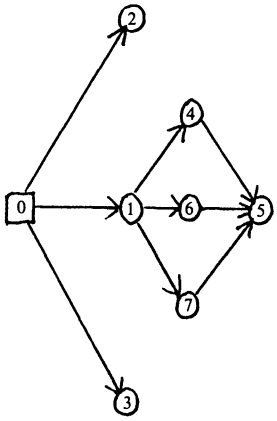

Figure 3. Graph of precedence relationships

$A_1 = \{(0, 1), (0, 2), (0, 3), (1, 4), (1, 6), (1, 7), (4, 5), (6, 5), (7, 5)\}$

TABLE 1

Matrix C

$n2$ / $n1$	0	1	2	3	4	5	6	7
0	—	0.00	0.00	0.00	—	—	—	—
1	—	—	1.00	2.00	.75	0.00	3.00	1.00
2	0.00	4.00	—	5.00	3.25	4.00	6.00	0.00
3	0.00	7.00	8.00	—	5.50	7.00	9.00	8.00
4	—	2.75	2.50	2.25	—	2.75	5.25	2.50
5	0.00	0.00	1.00	2.00	0.75	—	3.00	1.00
6	—	10.00	11.00	12.00	10.75	10.00	—	11.00
7	—	4.00	0.00	5.00	3.25	4.00	6.00	—

TABLA 2

Matrix P related to the graph shown in Figure 3

*n*2 / *n*1	0	1	2	3	4	5	6	7
0		1	1	1	1	1	1	1
1					1	1	1	1
2								
3								
4						1		
5								
6						1		
7						1		

TABLA 3

Matrix S related to Tables 1 and 2

*n*2 / *n*1	0	1	2	3	4	5	6	7
0		1	1	1				
1			1	1	1		1	1
2	1	1		1	1	1	1	1
3	1	1	1		1	1	1	1
4			1	1		1	1	1
5	1		1	1				
6			1	1	1	1		1
7			1	1	1	1	1	

In all problems, a lower bound, say $\underline{z}$ has been obtained by using the first three bounding procedures described in [1] for the ATSP. Our implementation is slightly different since, while obtaining $\underline{z}$, we consider a partial exploration of the precedence relationships by using the binary matrix S; see [3].

We obtain the initial FP by patching the cycles, if any of the optimal solution to the Assignment Problem (hereafter, AP) (2.6)-(2.9), so that the patches do not violate the precedence relationships; see [3]. Let r denote the cycle where the root node 0 is included; the other cycles are patched to cycle r in a given order. Two strategies are used for defining the order. Strategy 1 classifies the cycles according to the criterion of non-decreasing cardinality. Strategy 2 accepts the order that is offered by the AP algorithm (i.e., the order is defined ad random). The real-life problem of each class is solved by using the strategy 1; 50% of the generated problems use the strategy 1 and the others use the strategy 2.

Table 4 summarizes the results. The headings are as follows: OPT, number of cases where the algorithm detects the optimality of the FP. T (secs.), average CPU time for the TRIA and QUAD procedures. ntria and nquad, number of times the TRIA and QUAD have been used, respectively. GAP = $(\bar{z} - \underline{z})/\underline{z}$. average GAP: the two worst cases per each class are not included, nor the cases whose solution was proved to be optimal. We can see that the performance of the algorithm is very similar for the real-life and generated problems of each class. Note also that the CPU time increases almost linearly on the number of nodes; the range of the CPU time is very small within each class. Note also that

TABLE 4

Computational results (11 *entries per class*)

class	$\|N\|$	$\|A\|$	T (secs.)		OPT	GAP (%)		$T/\|N\| \times 100$	ntria	nquad
			average	max		average	max			
1	8	12	0.01	0.01	3	2.7	8.9	0.12	3	2
2	32	192	0.64	1.12	2	3.8	22.8	2.06	3	2
3	37	207	0.81	1.24	0	5.6	27.5	2.25	2	2
4	59	162	2.36	3.17	0	4.2	15.7	4.06	3	2
5	69	256	4.27	6.18	1	3.6	8.9	6.27	3	3
6	72	347	3.85	6.72	0	5.3	11.7	5.42	2	2
7	85	580	5.63	8.57	0	7.0	10.7	6.47	2	2
8	92	488	8.60	10.81	1	5.9	28.8	9.55	3	3
9	99	541	11.35	15.25	0	10.1	37.8	11.58	3	2

the maximum time required for solving any problem was 15.25 seconds. Only in one case the solution was be improved by hand; it amounted to 1.8% of the objective function value. As it was expected, the solution's optimality was proved only for a few cases. In any case, the gap between the FP and the lower bound on the optimal solution is not too-big.

Final remarks

The algorithm described in this paper does not attempt to guarantee the solution's optimality. Some cases may have a high GAP, it could mean that the FP is far away from the optimal solution, but we have detected by inspecting some small cases that the FP is optimal or near-optimal. High cost range and tight precedence relationships often produce high GAP. It seems that bounding procedures that use the information provided by the matrix P should be investigated; we are planning as future research to analyze the performance of using cutting planes LP based lower bounds (a good balance between the tightness of the lower bound and the computing effort is needed); identifying violated precedence relationships is not a big problem, but deciding what variables are to be free in the successive LP optimizations and what constraints are to be included as cutting planes is up to now an open problem. Other topic for future research consists of analyzing the performance of ATSP as a provider of a lower bound for our problem. In any case, based on our computational experience and the by-hand results, it seems that the proposed algorithm can be a useful tool for producing FP's almost on-line, as it is frequently required when planning e.g. the utilization of Flexible Manufacturing Systems.

ACKNOWLEDGEMENT

I wish to thank one of the referees for his careful reading of the paper; his comments have strongly improved the quality of the work. He also suggested to include the reference [9] and, thus, gave me the opportunity to point out the difference between the DARP and our ATSP-PR.

REFERENCES

[1] BALAS, E., and CHRISTOFIDES, N.: «A restricted Lagrangian approach to the Traveling Salesman Problem», *Mathematical Programming*, 21, 19-46, 1981.

[2] CAPANETO, G., and TOTH, P.: «Some new branching and bound criteria for the asymmetric travelling salesman problem», *Management Science*, 26, 736-743, 1980.

[3] ESCUDERO, L. F.: «An inexact algorithm for the sequential ordering problem», *European J. of Operation Research* (1988, in press).

[4] ESCUDERO, L. F., and PEREZ, G.: *A production planning problem in FMS*, Questiio, 11(1987)85-114.

[5] HOFFMAN, K., and PADBERG, M.: «LP-based combinatorial problem solving», *Annals of Operations Research*, 5, 145-194, 1986.

[6] KANELLAKIS, P. C., and PAPADIMITROU, Ch. C.: «Local search for the Asymmetric Travelling Salesman Problem», *Operations Research*, 28, 1086-1099, 1980.

[7] KUSIAK, A., and FINKE, G.: *Modeling and solving the flexible forging module scheduling problem, working paper* 85/06, Dept. of Mechanical and Industrial Engineering, University of Manitoba, Winnipeg, Canada, 1985.

[8] LAWLER, E. L.; LENSTRA, J. K.; RINNOOY-KAN, A. H. G., and SHMOYS, D. B. (eds.): *The Traveling Salesman Problem*, A guided tour of combinatorial optimization (Wiley, NY, 1985).

[9] PSARAFTIS, H. N.: «K-interchanges procedures for local search in a precedence-constrained routing problem», *European J. of Operational Research*, 13, 391-402, 1983.