RC 15597 (#69298) 3/19/90
Mathematics 16 pages

Research Report

FRAMEWORK OF AN LP CUTS-BASED ALGORITHM FOR THE SEQUENTIAL ORDERING PROBLEM WITH TIME WINDOWS AND PRECEDENCE RELATIONSHIPS

L.F. Escudero

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

A. Sciomachen

Universita degli Studi di Milano, Italy

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

FRAMEWORK OF AN LP CUTS-BASED ALGORITHM FOR THE SEQUENTIAL ORDERING PROBLEM WITH TIME WINDOWS AND PRECEDENCE RELATIONSHIPS

L.F. Escudero

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

A. Sciomachen

Universita degli Studi di Milano, Italy

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

1. PROBLEM DEFINITION

Let $\mathcal{H}$ denote a Hamiltonian path through a given set of nodes, say V, where $n \equiv |V|$. Let $i \rightarrow j$ mean that node i is immediately sequenced before node j in a given $\mathcal{H}$. Let $sp(i \rightarrow j)$ denote the ordered set of nodes from node i to node j in a given $\mathcal{H}$.

Let the digraph $G = (V,A)$ and the m-vector c be such that $c_{i,j}$ for $(i,j) \in A$ gives the weight associated with $i \rightarrow j$, and $m \equiv |A|$. Let the n-vector p be such that p_v gives the independent weight associated with node v for $v \in V$.

Let y_v denote the so-called accumulated potential ap of node v in a given $\mathcal{H}$. It can be expressed

$$y_v = p_\ell + \sum_{a \rightarrow b \in sp(\ell \rightarrow v)} T_{a,b} \tag{1.1}$$

where ℓ is the initial node in $\mathcal{H}$ and $T_{a,b} = c_{a,b} + p_b$. Let the n-vectors r and d be such that r_v (resp., d_v) gives the lower (resp., upper) bound on y_v for $v \in V$ in any feasible $\mathcal{H}$. Finally, let the arcset P give the precedence relationships among the nodes, such that $(i,j) \in P$ means that $y_i + T_{i,j} \leq y_j$ for any feasible $\mathcal{H}$.

The problem consists of finding a Hamiltonian path $\mathcal{H}$ with minimum weight in digraph G, such that the bounds on the ap of the nodes given by r and d are not violated and the precedence forcing constraints are satisfied.

The SOP has a broad application field. An obvious application is the ATSP with fixed city-origin and city-destination, where visiting each city requires a given amount of time and the arrival and departure should satisfy given dates. An interesting application arises in manufacturing planning. Basically, it is as follows. A single machine has to process a set of jobs; it can handle only one job at a time. The processing time of each job is given. There is a setup time between the processing of two consecutive jobs, and it depends on both jobs. (E.g., typically, a set of jobs with similar attributes is said to belong to the same type, provided that setup is only needed when changing job types). Each job may have a release date for its processing and a due date for its completion. Precedence relationships among the jobs are allowed. The goal consists of determining the processing sequence of the jobs in the machine such that release/due dates and precedence forcing constraints are satisfied, and the makespan (i.e., the completion time of the last job to be processed) is minimized.

This paper is organized as follows. Section 2 presents our 0-1 model for the SOP. Section 3 outlines the procedure for obtaining an optimal solution. It is based on an iterative tightening of the relaxations of the *Subtour Elimination Constraints* (SECs), *release and due Date Forcing Constraints* (DFCs) and *Precedence Forcing Constraints* (PFCs). Section 4 is devoted to the procedure for identifying violated SECs. The procedure has complexity $O(n^4)$. Section 5 deals with lifting the SECs such that a new set of valid inequalities can be appended to the LP relaxation of the problem in order to tighten the lower bound on the optimal solution.

2. The 0-1 MODEL

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

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

$$\min \; c^t x \tag{2.1}$$

subject to

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

$$x(\delta^-(v)) \; \leq \; 1 \text{ for all } v \in V \tag{2.3}$$

$$x(\delta^+(v)) \; \leq \; 1 \text{ for all } v \in V \tag{2.4}$$

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

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

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

$$m_{i,j}(1-\rho_{i,j}) \leq (y_i + T_{i,j}) - y_j \leq M_{i,j}(1-x_{i,j}) \quad \forall (i,j) \in A \tag{2.8}$$

$$y_i + T_{i,j} \leq y_j \text{ for all } (i,j) \in P \tag{2.9}$$

$$e_v \leq y_v \leq d_v \text{ for all } v \in V \tag{2.10}$$

where $x(F) = \sum_{(i,j) \in F} x_{i,j}$, $\delta^-(v) = \{(w,v) \in A\}$, $\delta^+(v) = \{(v,w) \in A\}$, $A(W) = \{(i,j) \in A | i,j \in W\}$
$e_v = r_v - 1 + p_v$,

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

$$\rho_{i,j} \equiv \begin{cases} x_{i,j} \text{ for } \gamma = 0 \\ 0 \text{ for } \gamma = 1 \end{cases} \tag{2.12}$$

and γ is a 0-1 parameter, such that $\gamma = 0$ for the case where $r_v = 1\ \forall v \in V$ and, otherwise, $\gamma = 1$.

We restrict c in (2.1) to the coordinates of A. Note that $z^* = y_\ell^* - p_\ell$, where z^* is the optimal solution of (2.1)-(2.10), and y_ℓ^* is the complexion time of the last node, say ℓ in the optimal sequence $\mathscr{H}^*$. Constraint (2.2) defines the number of arcs in any $\mathscr{H}$. Constraints (2.3) (resp., (2.4)) prevent that more than one node is sequenced immediately before (resp., after) any other node. Constraints (2.6) are the SECs (see e.g. [13]). Constraint (2.8) for (i,j) forces $y_j = y_i + T_{i,j}$ whenever $x_{i,j} = 1$; otherwise, it is not active. Note that (2.8) for $\gamma = 1$ together with (2.10) forces $max\ \{r_j - 1\ , y_i + c_{i,j}\} + p_j \le y_j \le d_j$ for $x_{i,j} = 1$, $(i,j) \in A$. Constraints (2.9) are the PFCs such that node i must be sequenced before node j for $(i,j) \in P$. (Note that the PFC (2.9) for $(i,j) \in P$ cannot be satisfied if $e_i + T_{i,j} > d_j$.) See in Ascheuer et al. [1] an equivalent formulation for enforcing the precedence relationships given by P in a non-release/due date environment. The *ap* of each node is lower and upper bounded by (2.10). The big enough values of $M_{i,j}$ and $m_{i,j}$ given in (2.11) do not cut-off any integer solution but are hopefully tight enough for the LP relaxation problem. Let (2.8) together with (2.10) be named DFCs. Note that (2.8) for $\gamma = 0$ is tighter than (2.8) for $\gamma = 1$.

An interesting experience could be to compare the computational effort required by problem (2.1)-(2.10), and problem *min* z subject to $y_v \le z$ for $\forall v \in V$ and constraints (2.2)-(2.10).

Note that (2.1)-(2.7) can be easily converted into the classical ATSP. Let the *Assignment-like Problem* (2.1)-(2.5) be named LPAP.

3. FRAMEWORK OF THE EXACT ALGORITHM

The basic methodology of the exact algorithm that we propose for solving (2.1)-(2.10) has the following main steps. (The overall framework is synthesized in the C-like procedure given in Figure 1.)

1. **Preprocessing-1**. Tightening the arcsets of candidate partial orderings (A) and precedence relationships (P). Note that arc (i,j) should be deleted from set A provided that it has been detected that $i \not\to j$ in any feasible $\mathscr{H}$.

A rough procedure is as follows.

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

$$e_i + T_{i,j} > d_j \tag{3.1}$$

$$\exists k \in V \text{ such that } x_{i,k} = 1 \text{ or } x_{k,j} = 1 \tag{3.2}$$

$$(j,i) \in P \tag{3.3}$$

b. Fix $x_{i,j} = 1$ provided that any of the following conditions is satisfied.

$$x_{k,j} = 0 \quad \forall k \in V \text{ such that } k \neq i \text{ and } \exists \ell | (\ell, j) \in P \quad \forall \ell \in V \tag{3.4}$$

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

c. Delete (i,j) from P provided that the following condition is satisfied

$$d_i + c_{i,j} < r_j \quad (i,j) \in P \tag{3.6}$$

Note that condition (3.1) implies $i \not\to j$ in any feasible $\mathcal{H}$. Condition (3.2) is implied by constraints (2.3) and (2.4). Condition (3.3) is implied by constraint (2.9). Condition (3.4) (resp., condition (3.5)) implies $i \to j$ in any feasible $\mathcal{H}$; notice that it states that node j (resp., node i) cannot be the initial (resp., final) node of any feasible $\mathcal{H}$ and all other potential relationships $k \to j$ (resp., $i \to k$) are not allowed. Note that (2.9) is redundant whenever condition (3.6) is satisfied.

2. **Preprocessing-2**. Reducing the gap $d_v - r_v$, $v \in V$. Notice that smaller gap, tighter DFCs constraints (2.8) and (2.10). In a recursive mode the vectors r and d should be updated as follows

$$r_b := max \{r_b , e_a + T_{a,b} + 1 \ \forall a \in V | (a,b) \in P\} \tag{3.7}$$

$$d_a := max \{d_a , d_b - T_{a,b} \ \forall b \in V | (a,b) \in P\} \tag{3.8}$$

3. **Initial upper bound** on the optimal solution. The approximate algorithm whose main steps are described in a companion paper (see [5]) is used to find the first upper bound, say $\bar{z}$ on the optimal solution.

4. **Initial lower bound** on the optimal solution. The first value of the lower bound, say, $\underline{z}$ is given by solving LPAP.

5. **Quasi-global problem selection and set-up**. Let the following set of variables

   a. The variables with value 1 in the solution provided by several runs of the approximate algorithm.

b. The basic variables in the optimal solution to LPAP.

c. The nonbasic variables in the optimal solution to LPAP with zero reduced cost (i.e., nonbasic alternative variables).

d. The nonbasic variables in the optimal solution to LPAP with value 1.

Note that this subset of variables is our current best estimation of the set of variables with value 1 in the optimal solution. Together with the constraints (2.2)-(2.5) and the objective function (2.1), this set of variables defines the first LP relaxation of the so-called quasi-global problem. Its initial solution to setup is the current best solution obtained so far. If no solution has been obtained by the approximate algorithm, the first version of the quasi-global problem will be the same global problem, and the set-up procedure is the standard one.

6. **LP quasi-global problem optimization**. Obtaining the optimal solution of the LP relaxation of the quasi-global problem.

7. **Pricing-out nonbasic variables**. To prove that the optimal solution to the LP quasi-global problem is also optimal to the LP global problem, one needs to price out the variables that were not included in the quasi-global problem.

8. **Enlarging the quasi-optimal problem**. If the reduced cost of any of these variables has not the appropriate sign, the quasi-global problem is revised by including the related variables. In this case the current LP solution is restored and go to Step 6. Formally, the quasi-global problem will be enlarged with the variables that satisfy condition (3.9).

$$\begin{aligned} &\bar{c}_{i,j} \le 0 \text{ for } \bar{x}_{i,j} = 0 \\ &\bar{c}_{i,j} \ge 0 \text{ for } \bar{x}_{i,j} = 1 \end{aligned} \tag{3.9}$$

where $\bar{x}_{i,j}$ denotes the value of the related variable in the LP optimal solution, and $\bar{c}_{i,j}$ is its reduced cost.

9. **Variables permanent fixing**. If the reduced costs $\{\bar{c}_{i,j}\}$ have the appropriate sign, the current LP optimal value, say $\underline{z}$, is a new (tighter) lower bound on the optimal solution to the global problem. In any case, the variable whose reduced cost satisfies condition (3.10) will be permanently fixed to its current value; note that it could be either zero or one.

$$|\bar{c}_{i,j}| \ge \bar{z} - \underline{z} \tag{3.10}$$

If, as a result of the testing, all nonbasic variables have been fixed then stop since we have obtained the optimal solution to the global problem.

10. If a substantial number of variables have been fixed in the previous step, then the phase **Pre-processing** is executed again.

11. **Quasi-global problem reduction** due to variables permanent fixing. It is obtained by:

    a. Detecting new *fixings by implication* and deleting the related fixed variables. (See conditions (3.1)-(3.6).)

    b. Updating the right-hand-side of constraint (2.2).

    c. Updating the vectors r and d. (See (3.7)-(3.8).)

    d. Deleting the constraint (2.3) (resp., (2.4)) related to node i (resp., j) for the permanent fixing $x_{ij} = 1$, if any.

    In case of any reduction that could affect the value of $\underline{z}$, the LP current solution is restored and go to Step 6. We must note that if the solution of two consecutive LP optimizations, say $\underline{z}^1$ and $\underline{z}^2$ is such that $\underline{z}^2 - \underline{z}^1 \leq \sigma$ (where σ is a positive tolerance, typically $\sigma = 10^{-4}$), then the reduced costs fixing and the constraints generation (see below) are not performed.

12. **Constraints generation**. Once it has been tested that no further problem reductions can be achieved with the above procedures, we proceed to the constraints identification phase, by appending to the quasi-global problem:

    a. The most violated SEC (Section 4) and lifted SEC (LSEC) (Section 5).

    b. The violated DFCs (2.8).

    c. The violated PFCs (2.9).

    By definition the complexity of the procedure for detecting violated DFCs and PFCs is $O(n^2)$. Note that if any of these constraints related to the pair (i,j) is appended to the quasi-global problem then the bounds (2.10) for i and j should also be appended.

    At the step where the new constraints are added, any non-active constraints previously appended is deleted from the current LP relaxation.

13. If the attempt is successful, the current LP solution is restored, and the LP dual-based optimization is continued from that basis by going to Step 6.

14. **Branch-and-cut phase**. If there is not any violated constraint except the integrality constraint (2.7), a branch-and-cut phase is executed in a similar way as described in [15]. At any

(*integer* or *non-integer*) node the procedure for constraints generation is executed, and violated constraints are appended to the related LP subproblem. Note that the new constraints will be *inherited* by the successors nodes; and then it could happen that integer nodes are converted into non-integer ones.

At each node with a new *incumbent* solution we analyze if a substantial number of nonbasic variables in the LP solution at node 1 can be permanently fixed at their current values. The testing is performed by using the reduced costs and the LP optimum at node 1, and the new incumbent solution as the (tighter) upper bound. If the attempt is successful, we fix the appropriate variables, proceed as in Step 10 and go to Step 11.

At selected non-integer nodes we proceed as follows:

a. Preprocessing to analyzing the feasibility of the partially fixed-and-branched subproblem attached to the node.

b. If it is detected that no successor node may provide a (better) feasible solution for the original problem, the node is fathomed.

c. Otherwise, an attempt to obtain a solution by using the approximate algorithm is performed, such that if a new incumbent is found we proceed as before.

The *branching node* is selected according to the criterion of the *best estimation* of the functional value; it is expressed as a linear combination of its functional value and the weighted sum of its integer infeasibilities, such that it can be written

$$F + \alpha \sum_{(i,j) \in A} min \{\bar{x}_{i,j}, 1 - \bar{x}_{i,j}\} \tag{3.11}$$

where F denotes the functional value at the given node, $\bar{x}_{i,j}$ gives the current value of variable $x_{i,j}$ at that node, and α is a parameter that can be expressed

$$\alpha = \frac{\bar{z} - \underline{z}}{\sum_{(i,j) \in A} \min \{\bar{x}^1_{i,j}, 1 - \bar{x}^1_{i,j}\}} \tag{3.12}$$

where $\bar{x}^1_{i,j}$ has the same meaning as above but, now, related to node 1 (i.e., the LP current quasi-global problem that provides the lower bound $\underline{z}$).

The parameter α is recomputed at each incumbent solution. In any case, *branching priority* is given to the nodes whose functional value is not worse than a given limit such that it can be

expressed by $\underline{b} + \beta$, where $\underline{b}$ gives the best functional value of the waiting nodes (note that $\underline{b} = \underline{z}$ at node 1), and β is a dynamically-adjusted parameter. In our case, $\beta = 0.1(\overline{z} - \underline{z})$ but it is gradually increased until every waiting node is a candidate for branching; at the same time that β is increased, higher priority is given to the nodes based on their integer infeasibility by increasing the parameter α. In fact, the value of β is increased by $0.1(\overline{z} - \underline{z})$ and α is doubled if υ consecutive nodes have been scanned without finding a new incumbent solution. The value of υ is a function of the number of variables, say ny, that take a positive value in the optimal solution of the current LP quasi-global problem, such that $\upsilon = 10 + 0.1ny$. See also [4].

15. To prove that the optimal solution to the quasi-global problem is also optimal to the global problem, the incumbent solution is set-up in the quasi-global problem as in Step 5.

16. Next, a testing is performed as in Step 7 by pricing-out the non-yet fixed variables that remain in the global problem. If the result of the testing is not successful, we proceed as in Step 8; the quasi-global problem will be included by:

    a. The set of variables that have not yet been deleted from the current version of the quasi-global problem.

    b. A set of variables from the global problem. At least, the set should be included by the variables that have not the appropriate sign on the reduced cost. In any case, some provisions have to be made to ensure that it is not an empty set so that the convergence is guaranteed.

17. Otherwise, the incumbent solution to the quasi-global problem is also optimal for the global problem.

We should emphasize the synergetic effect of combining the procedures for preprocessing obtaining initial feasible solutions, performing reduced cost fixing and implications, and generating violated constraints. See in different contests [3,4,10,12,13,15,16] among others. In [7] a different approach for getting tight lower bounds for the ATSP imbedded in the original problem is described. See related work in [2,6].

```
MAIN(SOP) {

  /* Fl = Flag for setting/revising the quasi_global_problem; */
  /* Diff = LP global optimum - LP quasi_global optimum;      */
  /* Numvar = Number of fixed variables;                      */

  Preprocess();
  Fl = TRUE;
  Do {
    Diff = Max;
    Do {
      If (Fl = = TRUE)
        Setup(Quasi_Global_Problem);
      Else
        Revise(Quasi_Global_problem);
      Diff = Solve(Quasi_Global_Problem);
      Fl = FALSE;
    }
    While (Diff != 0);
     Numvar = Fix_Variable();
    If (Numvar = = All)
      Optimum = TRUE;
    If (Numvar = = 0) {
      Constrset = Generate_Constrain();
      If (Constrset = = 0) {
        Branch&Bound();
        FL = TRUE;
      }
      Else
        Fl = FALSE;
    }
    If ((Numvar != 0) && (Numvar != All)) {
      Preprocess();
      Fl = TRUE;
    }
  }
  While (Optimum = = FALSE);
}


Preprocess() {
  Reduce(A);
  Compute(Upper_Bound);
  Compute(Lower_Bound);
}
```

Figure 1: C-like Procedure of the overall framework

4. IDENTIFYING VIOLATED SUBTOUR ELIMINATION CONSTRAINTS

The problem of identifying a violated subtour elimination constraint (2.6) is described in [1]. We outline here the procedure for completeness.

Let us assume that $\bar{x}$ is the optimal solution of problem (2.1)-(2.5) where some constraints (2.6), (2.8)-(2.10) may have been appended. We now consider the values $\bar{x}_{i,j}$ as capacities of the arcs $(i,j) \in A$. The goal consists of identifying a node set $W \subset V$ such that the related constraint (2.6) is a most violated SEC (if any). Then, the set W will be

$$W = arg.max\{\bar{x}(A(W')) - |W'| + 1 > 0\} \text{ for all } W' \subset V, 2 \leq |W'| \leq n-1. \tag{4.1}$$

For this purpose we will construct a symmetric directed graph, say $\overline{G}$, for which a certain *mincut* problem is to be solved. But, first, let us introduce additional notation. Let

$$\bar{\beta}_v = \bar{x}(\delta^-(v)) + \bar{x}(\delta^+(v)) \text{ for all } v \in V. \tag{4.2}$$

Next, we define an (auxiliary) digraph, say $G_0 = (V_0, A_0)$, as follows:

$$\begin{aligned} &V_0 = V \cup \{0\}, \text{ where 0 is a new node} \\ &A_0 = A \cup \{(0,v) \mid v \in V\} \cup \{(j,i) \mid (i,j) \in A \text{ and } (j,i) \notin A\}. \end{aligned} \tag{4.3}$$

(That is, we add to V a node 0 (the *source*), and we add to A arcs $(0,v)$ for all $v \in V$ and all reverse arcs but do not create parallel arcs). Let us define the following capacities for G_0

$$c^0_{0,v} := 1 - \frac{1}{2}\bar{\beta}_v \text{ for all } v \in V \tag{4.4}$$

(Note that $c^0_{0,v} \geq 0$ for all $v \in V$, where $c^0_{0,v} = 0$ whenever the related constraints (2.3) and (2.4) are satisfied as equalities). Further, we set

$$c^0_{i,j} := c^0_{j,i} := \frac{1}{2}(\bar{x}_{i,j} + \bar{x}_{j,i}) \text{ for all } (i,j) \in A_0 \backslash \{(0,v) \mid v \in V\} \tag{4.5}$$

(If $(i,j) \notin A$ (resp., $(j,i) \notin A$) we set $\bar{x}_{i,j} := 0$ (resp., $\bar{x}_{j,i} := 0$).)

Let $\delta_{\bar{0}}(W)$ denote the minimum capacity cut in $G_0 = (V_0, A_0)$ with respect to the capacities (4.4) and (4.5), such that $0 \notin W$, and let $c(\delta_{\bar{0}}(W))$ denote the associated capacity. It is possible to prove the following proposition.

Proposition 4.1:

If the following condition

$$c(\delta_0^-(W)) \geq 1 \tag{4.6}$$

is satisfied then there is no SEC violated by the current LP solution $\bar{x}$. *Otherwise, the SEC induced by* W *is a most violated constraint (2.6).*

Proof.

The capacity $c(\delta_{\bar{0}}(W))$ is just $|W| - \bar{x}(A(W))$. It can be shown as follows.

$$\begin{aligned} c(\delta_0^-(W)) &= \sum_{w \in W} c^0_{ow} + \sum_{v \in V \setminus W} \sum_{w \in W \mid (v,w) \in A_0} c^0_{vw} \\ &= |W| - 1/2 \sum_{w \in W} \bar{\beta}_w + 1/2 \sum_{a \in \delta(W)} \bar{x}_a \end{aligned} \tag{4.7}$$

where $\delta(W)$ is the arc set $\delta^-(W) \bigcup \delta^+(W)$ in the *original* digraph G being, then, $\delta^-(W) = \{(i,j) \in A | i \in V \setminus W, j \in W\}$ and $\delta^+(W) = \{(i,j) \in A | i \in W, j \in V \setminus W\}$

Note that the weight $\bar{x}_a$ for $a \equiv (i,j)$ with $i,j \in W$ is counted twice in the first sum of (4.7): first for i (in $\bar{\beta}_i$), then for j (in $\bar{\beta}_j$). It means that

$$\sum_{w \in W} \bar{\beta}_w = 2 \sum_{a \in A(W)} \bar{x}_a + \sum_{a \in \delta(W)} \bar{x}_a \tag{4.8}$$

Then,

$$c(\delta_0^-(W)) = |W| - \bar{x}(A(W)) \tag{4.9}$$

Therefore, $c(\delta_{\bar{0}}(W)) \geq 1 \Leftrightarrow \bar{x}(A(W)) \leq |W| - 1$. Note that for $|W| = 1$ it results that $c(\delta_{\bar{0}}(W)) = 1$. So, $c(\delta_{\bar{0}}(W)) < 1$ requires that $2 \leq |W|$ what is, precisely, our condition for activating constraint (2.6); then, the set W such that $c(\delta_{\bar{0}}(W)) < 1$ induces the most violated constraint (2.6). ■

We now construct a symmetric directed graph, say $\bar{G} := (V_0, \bar{A})$, related to digraph $G_0 = (V_0, A_0)$, by setting

$$\bar{A} := A_0 \bigcup \{(v,0) | v \in V\} \tag{4.10}$$

with capacities

$$\bar{c}_{i,j} := \bar{c}_{j,i} := c^0_{i,j} = c^0_{j,i} \text{ for all } (i,j) \in A_0 \setminus \{(0,v) | v \in V\} \tag{4.11}$$

$$\bar{c}_{v,0} := c^0_{0,v} \text{ for all } v \in V \tag{4.12}$$

($\overline{G}$ is a symmetric digraph since if any two nodes, say u and v, are linked by an arc then both arcs (u,v) and (v,u) are in the digraph). The underlying undirected graph $\hat{G} = (V_0,E)$ of $\overline{G} = (V_0, \overline{A})$ is defined by

$$E = \{ ij \,|(i,j) \in \overline{A}\} \tag{4.13}$$

with capacities

$$\hat{c}_{ij} := \bar{c}_{i,j} = \bar{c}_{j,i} \text{ for all } ij \in E \tag{4.14}$$

It has the property that for each $W \subset V$,

$$\hat{c}(\delta(W)) = c(\delta_0^-(W)) \tag{4.15}$$

where $\delta(W) = \{ ij \in E | i \in V \backslash W, j \in W\}$.

Thus, a minimum capacity cut $\delta(W)$ in $\hat{G}$ corresponds to a minimum capacity cut $\delta_0^-(W)$ in G_0 and vice versa. Such a cut $\delta(W)$ can be obtained with the Gomory-Hu procedure. Padberg and Rinaldi [14] describe a practically efficient version of this procedure. The main loop of the algorithm requires at most n major iterations. At any major iteration two nodes at least are "contracted" or "shrunk" into a single node. Two main steps are executed at each major iteration: first, a series of tests are designed to shrink as many nodes as possible; and, second, a *maxflow* algorithm is executed by selecting any two nodes of the shrunk graph as nodes *source* and *sink*. The algorithm described by Goldberg and Tarjan [8] is used in the second step; its complexity is $p(n) = O(mn \log(n^2/m))$, where $m \equiv |E|$. The tests that are performed in the first step have a complexity not worse than $p(n)$. Then, the overall complexity is $O(n^4)$. Although this is not better than the complexity of other published algorithms, the "shrinking" mechanism suggests a better efficiency in practice; this is confirmed by extensive computational experience reported in [15,16]. Since the algorithm obtains successively a series of better cuts till an optimal one is reached, we may generate the related induced SECs if they are violated, instead of using only a 'best' one.

5. IDENTIFYING VIOLATED LIFTED SUBTOUR ELIMINATION CONSTRAINTS

The current LP solution may not violate any SEC, but it may violate other types of constraints that implicitly are satisfied by any (integer) feasible solution to the original problem. Various general classes of inequalities that are valid for the ATSP are determined in [6,9,11]; then, they are also valid for our problem. This type of inequalities is obtained by lifting k-cycles. In partic-

ular, let the so-called k-Lifted Subtour Elimination Constraint (LSEC) types a and b be expressed by (5.1) and (5.2), respectively for the ordered set of nodes $\{i_1, i_2, ..., i_k\}$, $3 \le k < n-1$.

$$\sum_{l=1}^{k-1} x_{i_l, i_{l+1}} + x_{i_k, i_1} + 2\sum_{l=2}^{k-1} x_{i_l, i_1} + \sum_{l=3}^{k-1}\sum_{h=2}^{l-1} x_{i_l, i_h} \le k-1 \tag{5.1}$$

$$\sum_{l=1}^{k-1} x_{i_l, i_{l+1}} + x_{i_k, i_1} + 2\sum_{l=3}^{k} x_{i_1, i_l} + \sum_{l=4}^{k}\sum_{h=3}^{l-1} x_{i_l, i_h} \le k-1 \tag{5.2}$$

Figure 2 shows the elements of $\bar{x}$ that are used by (5.1)-(5.2).

Let the 4-LSEC types c and d be expressed by (5.3) and (5.4) for the ordered set of nodes $\{i_1, i_2, i_3, i_4\}$, respectively.

$$\sum_{l=1}^{3} x_{i_l, i_{l+1}} + x_{i_4, i_1} + 2x_{i_2, i_1} + x_{i_2, i_4} + x_{i_3, i_1} + x_{i_4, i_3} \le 3 \tag{5.3}$$

$$\sum_{l=1}^{3} x_{i_l, i_{l+1}} + x_{i_4, i_1} + 2x_{i_1, i_3} + 2x_{i_3, i_1} \le 3 \tag{5.4}$$

The identification in reasonable polynomial time of the most violated k-LSEC for any k is an open problem up to now. In any case, note that the complexity is $O(n^k)$ if an exhaustive search is performed.

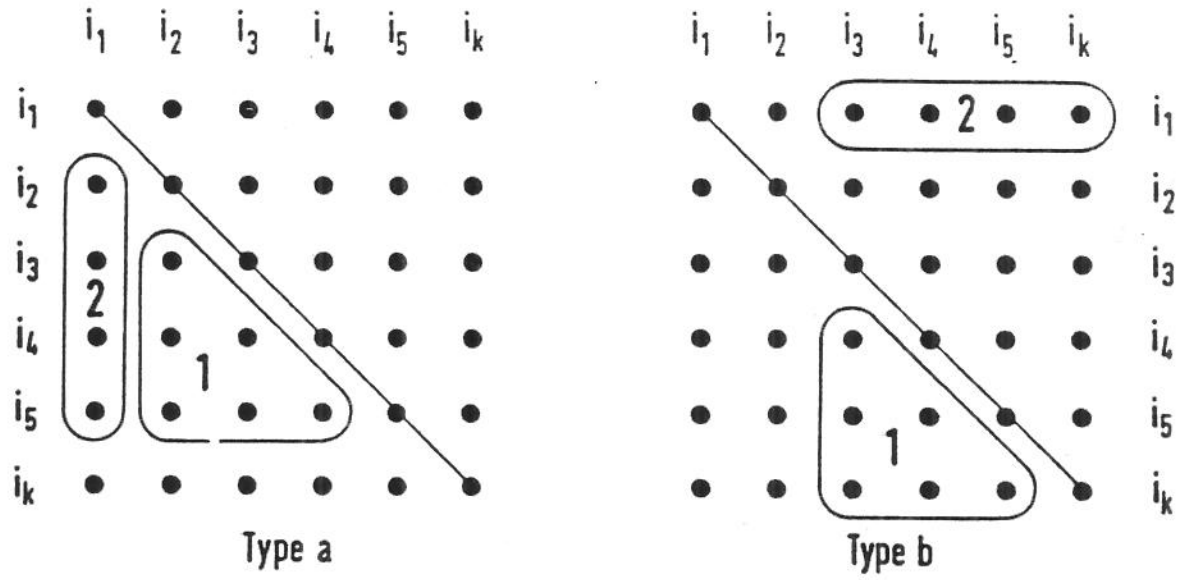

Figure 2. Elements of $\bar{x}$ to be used by k-LSECs

We will use k-LSECs only for $k=3$ and 4. Due to computational effort considerations, we do not analyze all constraints for $k=4$. In fact, constraints (5.1)-(5.4) are not tested for the ordered set of nodes $\{i_1, i_2, i_3, i_4\}$ if, at least, one of the following conditions is satisfied.

$$\bar{x}_{i_1,i_2} + \bar{x}_{i_2,i_3} + 2\bar{x}_{i_2,i_1} + 2\bar{x}_{i_3,i_1} + \bar{x}_{i_3,i_2} \leq \eta_1 \tag{5.5}$$

$$\bar{x}_{i_1,i_2} + \bar{x}_{i_2,i_3} + 2\bar{x}_{i_1,i_3} \leq \eta_2 \tag{5.6}$$

$$\bar{x}_{i_1,i_2} + \bar{x}_{i_2,i_3} + 2\bar{x}_{i_2,i_1} + \bar{x}_{i_3,i_1} \leq \eta_3 \tag{5.7}$$

$$\bar{x}_{i_1,i_2} + \bar{x}_{i_2,i_3} + 2\bar{x}_{i_1,i_3} + 2\bar{x}_{i_3,i_1} \leq \eta_4 \tag{5.8}$$

Note that the target is to avoid a complexity of $O(n^4)$ under the assumption that if any of the conditions (5.5)-(5.8) is satisfied then the related 4-LSEC is not likely to be violated. (We recommend the following values: $\eta_1 = 2.1$, $\eta_2 = 1.2$, $\eta_3 = 1.5$ and $\eta_4 = 1.8$.)

REFERENCES

[1] N. Ascheuer, L.F. Escudero, M. Groetschel, and M. Stoer, On identifying in polynomial time violated subtour elimination and precedence forcing constraints for the sequential ordering problem, Integer Programming and Combinatorial Optimization Conference, Waterloo (Canada), May 1990.

[2] E. Balas and M. Fischetti, A lifting procedure for the asymmetric traveling salesman polytope and a large new class of facets, MSRR-556, GSIA, Carnegie Mellon University, Pittsburg, PA, 1989.

[3] H. Crowder and M. Padberg, Solving large-scale symmetric traveling salesman problems to optimality, **Management Sciences** 26 (1980) 495-509.

[4] L.F. Escudero, A production planning problem in FMS, **Annals of Operations Research** 17 (1989) 69-104.

[5] L.F. Escudero and A. Sciomachen, An approximate algorithm for the sequential ordering problem with time windows and precedence relationships (1990, in preparation).

[6] M. Fischetti, Facets of the asymmetric traveling salesman polytope, **Mathematics of Operations Research** (1990, to appear).

[7] M. Fischetti and P. Toth, An additive bounding procedure for combinatorial optimization problems, **Operations Research** 37 (1989) 319-328.

[8] A.V. Goldberg and R.E. Tarjan, A new approach to the maximum flow problem, **Journal of ACM** 35 (1988) 921-940.

[9] M. Groetschel, Polyedrische Charakterisierungen kombinatorischer Optimierungsprobleme (Hain, Meisenheim am Glan, 1977).

[10] M. Groetschel, M. Juenger and G. Reinelt, A cutting plane algorithm for the linear ordering problem, **Operations Research** 34 (1984) 1195-1220.

[11] M. Groetschel and M. Padberg, Lineare Charakterisierungen von Travelling Salesman Problemen, **Z. Operations Research** 21 (1977) 33-64.

[12] K. Hoffman and M. Padberg, LP-based combinatorial problem solving, **Annals of Operations Research** 5 (1986) 145-194.

[13] M. Padberg and M. Groetschel, Polyhedral computations, in: E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy-Kan and D.B. Shmoys (eds.), The Traveling Salesman Problem, A guided tour of combinatorial optimization (Wiley, NY, 1985) 251-360.

[14] M. Padberg and G. Rinaldi, An an efficient algorithm for the minimum capacity cut problem, Preprint, New York University, NY, 1987.

[15] M. Padberg and G. Rinaldi, Optimization of a 532-city symmetric traveling salesman problem, **Operations Research Letters** 6 (1987) 1-7.

[16] M. Padberg and G. Rinaldi, A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems, num. 319, Laboratoire d'Econometrie de l'Ecole Polytechnique, Paris, 1989.