J. E. DENNIS, JR. AND J. J. MORÉ (1974), *A characterization of superlinear convergence and its application to quasi-Newton methods*, Math. Comp., 28, pp. 549–560.

——(1977), *Quasi-Newton methods, motivation and theory*, SIAM Rev., 19, pp. 46–89.

J. E. DENNIS, JR. AND R. B. SCHNABEL (1983), *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, NJ.

A. V. FIACCO AND G. P. MCCORMICK (1968), *Nonlinear Programming*, John Wiley and Sons, New York.

R. FLETCHER (1980), *Practical Methods of Optimization*, John Wiley and Sons, New York.

R.-P. GE AND M. J. D. POWELL (1983), *The convergence of variable metric matrices in unconstrained optimization*, Math. Programming, 27, pp. 123–143.

P. E. GILL, W. MURRAY, AND M. H. WRIGHT (1981), *Practical Optimization*, Academic Press, London.

D. GOLDFARB (1970), *A family of variable metric methods derived by variational means*, Math. Comp., 24, pp. 23–26.

M. D. HEBDEN (1973), *An algorithm for minimization using exact second derivatives*, Report TP515, A.E.R.E., Harwell, England.

J. J. MORÉ (1977), *The Levenberg–Marquardt algorithm: Implementation and theory*, in Numerical Analysis, G. A. Watson, ed., Lecture Notes in Math. 630, Springer-Verlag, Berlin, pp. 105–116.

J. J. MORÉ, B. S. GARBOW, AND K. E. HILLSTROM (1981), *Testing unconstrained optimization software*, ACM Trans. Math. Software, 7, pp. 17–41.

J. J. MORÉ AND D. C. SORENSEN (1983), *Computing a trust region step*, SIAM J. Sci. Statist. Comput., 4, pp. 553–572.

M J. D. POWELL (1976), *Some global convergence properties of a variable metric algorithm for minimization without exact line searches*, in Nonlinear Programming, SIAM-AMS Proceedings, Vol. IX, R. W. Cottle and C. E. Lemke, eds., Society for Industrial and Applied Mathematics, Philadelphia.

R. B. SCHNABEL, J. E. KOONTZ, AND B. E. WEISS (1982), *A modular system of algorithms for unconstrained minimization*, ACM Trans. Math. Software, 11, pp. 419–440.

SIAM J. OPTIMIZATION
Vol. 3, No. 1, pp. 25–42, February 1993

A CUTTING PLANE APPROACH TO THE SEQUENTIAL ORDERING PROBLEM (WITH APPLICATIONS TO JOB SCHEDULING IN MANUFACTURING)*

N. ASCHEUER†, L. F. ESCUDERO‡, M. GRÖTSCHEL†, AND M. STOER†

Abstract. The sequential ordering problem (SOP) finds a minimum cost Hamiltonian path subject to certain precedence constraints. The SOP has a number of practical applications and arises, for instance, in production planning for flexible manufacturing systems. This paper presents several 0–1 models of the SOP and reports the authors' computational experience in finding lower bounds of the optimal solution value of several real-life instances of SOP. One of the most successful approaches is a cutting plane procedure that is based on polynomial time separation algorithms for large classes of valid inequalities for the associated polyhedron.

Key words. traveling salesman problem, sequential ordering problem, linear ordering problem, precedence constraints, cutting plane algorithm, separation algorithm, polyhedral combinatorics

AMS(MOS) subject classifications. 90C10, 90C35

1. Introduction and problem definition. Problems for the flexible manufacturing systems we are considering (see, e.g., [6]) can be phrased in graph theoretical terminology in the following way. We are given a directed or undirected graph where an arc or edge represents the possibility of performing two tasks consecutively and where a (e.g., transportation or set-up) cost is incurred by changing from one task to another. In addition, some precedence relations are given that specify that some tasks have to be executed before certain others. The problem is to schedule all jobs at minimum cost, i.e., to find a feasible Hamiltonian path, say $\mathcal{H}$ of minimum cost, where $\mathcal{H}$ is called *feasible* if it does not violate the precedence constraints.

In this paper (and in the real application that motivated this work) the given graph is the complete directed graph $D_n = (V, A_n)$ on n nodes. (An application, where the given graph is undirected, can be found in [22].) We denote an arc going from some node i to another node j by (i, j) and the associated cost by c_{ij}. The precedence constraints are given by a digraph $P = (V, R)$, on the same node set V as D_n, where an arc $(i, j) \in R$ means that task i has to be performed before task j. Clearly, this *precedence digraph* P has to be acyclic (i.e., may not contain a directed cycle). Moreover, if (i, j), $(j, k) \in R$ then k cannot be performed before i; in other words, we can also assume that P is transitively closed.

So the precedence constraints are given by an acyclic and transitively closed digraph $P = (V, R)$. Using this notation we call a Hamiltonian path in D_n *feasible* if $(j, i) \notin R$ holds for all $i < j$, where $i < j$ means that there is a directed path from node i to node j in the Hamiltonian path.

Now we can state the *sequential ordering problem* (SOP) formally. Given a complete digraph $D_n = (V, A_n)$ with costs c_{ij} for all $(i, j) \in A_n$ and a transitively closed acyclic digraph $P = (V, R)$, find a feasible Hamiltonian path $\mathcal{H}$ in D_n that has minimum cost.

If the precedence digraph $P = (V, R)$ has empty arc set, the SOP reduces to finding a minimum cost Hamiltonian path in D_n. This is an NP-hard problem and so is the SOP. Our main concern here, though, is not an algorithm for the "pure" Hamiltonian

* Received by the editors December 26, 1990; accepted for publication (in revised form) November 13, 1991.

† Institut für Mathematik, Universität Augsburg, Germany.

‡ Facultad de Matematicas, Universidad Complutense de Madrid, Spain.

path problem (or, equivalently, the asymmetric traveling salesman problem, ATSP) but a method that deals with precedences.

This paper is organized as follows. In § 2 we present three different 0–1 models of the SOP, in particular, some classes of inequalities valid for the associated polyhedra. Polynomial time *separation algorithms* for some of these classes are described in § 3. Further classes of valid inequalities are discussed in § 4. In § 5 we present some preprocessing procedures for our cutting plane algorithm that help reduce the instance sizes. The implementations of the cutting plane algorithm are outlined in § 6; our computational results are reported in § 7.

2. 0–1 Models. The SOP, in the form stated here, seems to have been formulated for the first time in [4]. The aim of [4] and the subsequent paper [5] was the design of a heuristic that performs well in practice with respect to running time and solution quality. It was decided, however, to analyze the quality performance of the heuristic before using its implementation in a production planning system.

Before describing the 0–1 model of the SOP introduced in [4], we introduce the following notation.

Let $D_n=(V, A_n)$ be the complete digraph of n nodes and let $P=(V, R)$ be a transitively closed, acyclic subdigraph of D_n. We set

$$\tilde{R} := \{(j, i) \in V \times V \mid (i,j) \in R\}, \tag{2.1a}$$

$$\vec{R} := \{(i, k) \in V \times V \mid \exists j \text{ with } (i,j), (j,k) \in R\}, \tag{2.1b}$$

$$A := A_n \setminus (\vec{R} \cup \tilde{R}). \tag{2.1c}$$

Note that a feasible Hamiltonian path can contain neither an arc from $\tilde{R}$ nor an arc from $\vec{R}$, while for each arc in A there is some feasible Hamiltonian path containing this arc. We thus call A the *feasible arc set* and $D=(V, A)$ the *feasible subdigraph* of D_n. Furthermore, set

$$\alpha_k := 1+|\{i \mid (i,k) \in R\}|, \qquad k \in V, \tag{2.2a}$$

$$\beta_k := n-|\{j \mid (k,j) \in R\}|, \qquad k \in V. \tag{2.2b}$$

It is clear that α_k-1 (respectively, $n-\beta_k$) is the minimum number of predecessor (respectively, successor) nodes for node k in any feasible Hamiltonian path.

Let us introduce the following two types of variables. For each arc $(i,j) \in A$, x_{ij} is a 0–1 variable that indicates whether (i,j) is in the Hamiltonian path (i.e., $x_{ij}=1$) or not. (We do not need variables for the arcs from $A_n \setminus A$.) The second type are 0–1 variables ξ_{kh} for $k, h \in V$, which are auxiliary variables that help to model the precedence constraints, such that $\xi_{kh}=1$ means that node k is to be sequenced at level h for $\alpha_k \leqq h \leqq \beta_k$ and, otherwise, zero.

To obtain a compact formulation we introduce further terminology. If F is a subset of A we abbreviate the sum $\sum_{(i,j) \in F} x_{ij}$ by $x(F)$. If W is a subset of V then $A(W)=\{(i,j) \in A \mid i,j \in W\}$. If $j \in V$ then $\delta^+(j)=\{(j,k) \in A\}$ and $\delta^-(j)=\{(i,j) \in A\}$, and if, moreover, $W \subseteq V \setminus \{j\}$ then $(j:W)=\{(j,k) \in A \mid k \in W\}$ and $(W:j)=\{(i,j) \in A \mid i \in W\}$. Let us now assume that, in addition to $D_n=(V, A_n)$ and $P=(V, R)$, costs $c_{ij} \in \mathbb{R}$ for all $(i,j) \in A$ are given.

The model introduced in [4] is as follows.

$$\lambda^* = \min c'x \quad \text{subject to} \tag{2.3}$$

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

$$x(\delta^-(j)) \leqq 1 \quad \text{for all } j \in V, \tag{2}$$

$$x(\delta^+(j)) \leqq 1 \quad \text{for all } j \in V, \tag{3}$$

$$x_{ij} \geqq 0 \quad \text{for all } (i,j) \in A, \tag{4}$$

$$x(A(W)) \leqq |W|-1 \quad \text{for all } W \subset V, \quad 2 \leqq |W| \leqq n-1, \tag{5}$$

$$x_{ij} \in \{0; 1\} \quad \text{for all } (i,j) \in A, \tag{6}$$

$$\sum_{k \mid \alpha_k \leqq h \leqq \beta_k} \xi_{kh} = 1 \quad \text{for all } h \in V, \tag{7}$$

$$\sum_{\alpha_k \leqq h \leqq \beta_k} \xi_{kh} = 1 \quad \text{for all } k \in V, \tag{8}$$

$$\sum_{\alpha_i \leqq h_i \leqq \beta_i} h_i \xi_{ih_i} + 1 \leqq \sum_{\alpha_j \leqq h_j \leqq \beta_j} h_j \xi_{jh_j} \quad \text{for all } (i,j) \in R \setminus \vec{R}, \tag{9}$$

$$\xi_{ih} + \xi_{jh+1} \leqq 1 \quad \text{for all } (i,j) \in A_n \setminus A, \quad \max\{\alpha_i, \alpha_j-1\} \leqq h \leqq \min\{\beta_i, \beta_j-1\}, \tag{10}$$

$$\xi_{kh} \in \{0; 1\} \quad \text{for all } k \in V, \quad \alpha_k \leqq h \leqq \beta_k, \tag{11}$$

$$\xi_{ih} + \xi_{jh+1} \leqq 1 + x_{ij} \quad \text{for all } (i,j) \in A, \quad \max\{\alpha_i, \alpha_j-1\} \leqq h \leqq \min\{\beta_i, \beta_j-1\}. \tag{12}$$

We briefly indicate the logic of the model. In analogy to the well-known 0–1 model of the ATSP, constraints (1)–(6) provide an IP-formulation of the Hamiltonian path problem in $D=(V, A)$. Inequalities (5) are called, as usual, *subtour elimination constraints* (SECs).

Constraints (7)–(12) ensure that the given precedence constraints are observed. Constraints (7) (respectively, constraints (8)) force one node (respectively, level) per level (respectively, node). Constraints (9) prevent reverse sequencing for pairs of nodes that are linked by *direct* precedence relationships. Constraints (10) prevent illegal immediate sequencings. Finally, constraints (12) are the so-called *linking constraints* that integrate submodels (1)–(6) and (7)–(11).

Clearly, there are a number of model improvements possible, e.g., turning some of the inequalities into equalities, etc., but we state here only the basic model.

The computational experience with this model reported in [4] and [5] was unsatisfactory with respect to the integrality gap, i.e., in a number of cases the relative deviation $(\lambda^H-\lambda_{LR})/\lambda_{LR}$ was rather large, where λ^H gives the cost of the solution found by the heuristic algorithm, and λ_{LR} is the lower bound of the optimal value λ^* obtained from the (restricted) Lagrangian relaxation of model (2.3) used in [4]. Such a gap has one of the following causes. Either λ^H or λ_{LR}, or both, are far away from λ^*. The belief was that λ^H was good and λ_{LR} poor. This belief motivated the introduction and investigation of further 0–1 models of the SOP, which is the subject of the rest of the paper.

The first new model requires two types of variables. The first type are 0–1 variables x_{ij} with the same meaning as before. The second type are real variables y_{ij} for all $(i,j) \in A_n$, which are auxiliary variables that help to model the precedence constraints.

Our first new model of the SOP is as follows.

$$\lambda^* = \min c'x \quad \text{subject to} \quad x \text{ satisfies (2.3) (1)–(6) and} \tag{2.4}$$

$$y_{ij}=1 \quad \text{for all } (i,j) \in R, \tag{7}$$

$$y_{ij}+y_{ji}=1 \quad \text{for all } (i,j) \in A_n, \tag{8}$$

$$y_{ij}+y_{jk}+y_{ki} \leqq 2 \quad \text{for all } i,j,k \in V, \quad i \neq j \neq k, \tag{9}$$

$$y_{ij} \geqq 0 \quad \text{for all } (i,j) \in A_n, \tag{10}$$

$$x_{ij}-y_{ij} \leqq 0 \quad \text{for all } (i,j) \in A. \tag{11}$$

If we add integrality constraints to (8)–(10), we obtain a well-known 0-1 formulation of the linear ordering problem; see [10]. In our case, integrality stipulations for the y_{ij}'s are not needed, since integrality of the x_{ij}'s implies integrality of the y_{ij}'s via (11). Constraints (7)–(11) ensure that the given precedence constraints are observed. Clearly, there are a number of model improvements possible, e.g., we can also skip some of the variables y_{ij}'s, turn some of the inequalities to equalities (see § 5), etc. These obvious modifications have been done in our implementation. We state here only the basic model for notational ease.

A nice feature of model (2.4) is that it combines two well-known combinatorial optimization problems in a natural way. Looking at this model we can say that the SOP is the Hamiltonian path problem plus the linear ordering problem integrated through the *linking constraints* (11). An obvious disadvantage of this model is the use of the auxiliary variables y_{ij}'s. In fact, we can get rid of these by replacing (7)–(11) by a new class of constraints of size exponential in n.

Our second new model of the SOP is as follows.

$$\lambda^* = \min c'x \quad \text{subject to} \quad x \text{ satisfies (1)–(6) and} \tag{2.5}$$

$$x((j: W)) + x(A(W)) + x((W: i)) \leqq |W| \tag{12}$$

$$\text{for all } (i,j) \in R \quad \text{and all } \emptyset \neq W \subseteq V \setminus \{i,j\}.$$

We call the inequalities (12) *precedence forcing constraints* (PFCs). It is obvious that every feasible solution of (1)–(6) and (12) is the incidence vector of a feasible Hamiltonian path and vice versa.

Although model (2.4) provides a nice interpretation of the SOP as a combination of two other well-known problems, our computational experience (see below) shows that model (2.5) is a more natural setting for the SOP, given the type of separation algorithms that we propose.

Both models give rise to polyhedra associated with the SOP. We only introduce here the one arising from (2.5). Let $D_n = (V, A_n)$ be the complete digraph on n nodes, let $P = (V, R)$ be a transitively closed acyclic subdigraph of D_n, $A := A_n \setminus (\vec{R} \cup \tilde{R})$, and set

$$\text{SOP}(n, P) := \text{conv}\{x \in \mathbb{R}^A \mid x \text{ satisfies (1)–(6), (12)}\}. \tag{2.6}$$

SOP (n, P) is called the *sequential ordering polytope* associated with D_n and P, since every point that satisfies (1)–(6) and (12) is an incidence vector of a feasible Hamiltonian path, i.e., a feasible solution of the SOP. The study of the structure of this polytope (dimensions, facets, etc.) is of course of particular interest for the solution of the SOP. It is clearly closely related to the study of the ATSP polytope; see [16]. The scope of the present paper is, however, computational and there is no space here to discuss even some of the basic polyhedral facts about SOP (n, P).

3. Separation algorithms. The cutting plane algorithms we are going to describe follow the standard scheme described in, e.g., [3], [8]–[10], [13], [16], [20], and [21]. One of the main ingredients of such an algorithm are routines that check whether a given point (usually the optimum solution of the last LP relaxation solved) satisfies all inequalities of some given class of constraints and, if not, output at least one inequality of this class violated by the given point.

Such procedures are called *separation algorithms*; see [11] for some theory behind this approach. Of course, we are interested in separation algorithms that run in polynomial time.

In this section we describe polynomial time separation algorithms for the subtour elimination constraints (SECs) (2.3) (5) and the precedence forcing constraint (PFCs) (2.5) (12); see also [1]. Note that both classes contain a number of inequalities that is exponential in n. (Note also that constraints (1)–(6) of model (2.3) are inherited by models (2.4) and (2.5).)

We begin with the SECs. The input of our separation algorithm is a point $z \in \mathbb{Q}^A$. We assume that $z_{ij} \geqq 0$ for all $(i,j) \in A$; we do not require that z satisfies constraints (1)–(3) of (2.3), i.e., our algorithm will handle more general situations than those arising in models (2.3), (2.4), and (2.5). The output of the algorithm provides either the statement that z satisfies all inequalities

$$x(A(W)) \leqq |W| - 1 \quad \text{for all } W \subseteq V, \quad 2 \leqq |W| \leqq n, \tag{3.1}$$

or it provides a node set $W \subseteq V, 2 \leqq |W| \leqq n$ such that $z(A(W)) > |W| - 1$. In fact (this will be clear from the description of the algorithm), we can even find a node set W such that $z(A(W)) - |W| + 1$ is as large as possible, i.e., a *most violated* SEC can be identified.

For this purpose we construct a (first) auxiliary digraph $D_0 = (V_0, A_0)$ as follows.

$$V_0 = V \cup \{0\}, \quad \text{where 0 is a new node,} \tag{3.2a}$$

$$A^z := \{(i,j) \in A \mid z_{ij} > 0\}, \tag{3.2b}$$

$$A_0 := A^z \cup \{(0,v) \mid v \in V\} \cup \{(j,i) \mid (i,j) \in A^z \text{ and } (j,i) \notin A^z\}. \tag{3.2c}$$

In other words, we make $D^z = (V, A^z)$ symmetric by reversing arcs and add a source zero that is linked to all nodes in D^z. We solve the separation problem for the SECs (5) by reducing it to a sequence of min-cut problems. To do this we introduce (auxiliary) capacities c^0_{ij} for the arcs of D_0 in the following way. First, we set

$$\zeta_j = z(\delta^-(j)) + z(\delta^+(j)) \quad \text{for all } j \in V \tag{3.3}$$

and we define the capacities c^0_{ij} by

$$c^0_{0j} := 1 - \tfrac{1}{2}\zeta_j + M \quad \text{for all } j \in V, \tag{3.4}$$

where M is a positive number chosen such that $c^0_{0j} \geqq 0$ for all $j \in V$. Furthermore, we set

$$c^0_{ij} := c^0_{ji} := \tfrac{1}{2}(z_{ij} + z_{ji}) \quad \text{for all } (i,j) \in A^z. \tag{3.5}$$

(In case $(j,i) \notin A^z$ for some $(i,j) \in A^z$ we assume z_{ij} to have value zero.)

Now we introduce n further auxiliary digraphs that are slight modifications of D_0 as follows. For every $k \in V$ we define a digraph $D_k = (V_k, A_k)$ with capacities c^k_{ij} by setting

$$V_k := V_0, \tag{3.6a}$$

$$A_k := A_0 \dot{\cup} B_k \quad \text{where } B_k = \{(v,k) \mid v \in V \setminus \{k\}\}, \tag{3.6b}$$

$$c^k_{ij} := c^0_{ij} \quad \text{for all } (i,j) \in A_0, \tag{3.6c}$$

$$c^k_{vk} := M \quad \text{for all } (v,k) \in B_k. \tag{3.6d}$$

(In (b) above $\dot{\cup}$ means disjoint union, i.e., if A_0 contains an arc from B_k, we add a parallel one.)

(3.7) Separation Algorithm for the Subtour Elimination Constraints.

Input. A point $z \in \mathbb{Q}^A$ satisfying $z_{ij} \geqq 0$.

Output. At least one node set of cardinality between 2 and n, such that the corresponding SEC is violated by z, or the information that no such node set exists.

For each $k \in V$ do:

1. Construct the digraph $D_k = (V_k, A_k)$ with capacities c_{ij}^k as outlined before; see (3.6).
2. Use a max-flow algorithm to determine a $(0, k)$-cut $\delta^-(W_k)$ in D_k (i.e., a cut separating 0 and k such that $k \in W_k$, $0 \notin W_k$), so that its capacity $c^k(\delta^-(W_k))$ is as small as possible.
3. If $c^k(\delta^-(W_k)) < nM+1$ then $x(A(W_k)) \leqq |W_k|-1$ is a SEC violated by z.

End For

If the above procedure does not output a violated constraint then z satisfies all SECs.

LEMMA 3.8. *If, for all $k \in V$, the minimum capacity of a $(0, k)$-cut in D_k is not smaller than $nM+1$, then z satisfies all inequalities $z(A(W)) \leqq |W|-1$, $W \subseteq V$, $2 \leqq |W| \leqq n$. If, for some $k \in V$, there is a $(0, k)$-cut $\delta^-(W_k)$, $W_k \subseteq V$, $2 \leqq |W_k| \leqq n$ with $c^k(\delta^-(W_k)) < nM+1$, then $z(A(W_k)) > |W_k|-1$.*

Proof. The capacity $c^k(\delta^-(W_k))$ of any cut $\delta^-(W_k)$ in D_k with $0 \notin W_k$, $k \in W_k$ is nothing but $|W_k| - z(A(W_k)) + nM$. This can be seen as follows.

$$
\begin{aligned}
c^k(\delta^-(W_k)) &= \sum_{w \in W_k} c_{0w}^k + \sum_{v \in V \setminus W_k} \sum_{w \in W_k \mid (v,w) \in A_n} c_{vw}^k + \sum_{v \in V \setminus W_k \mid (v,k) \in B_k} c_{vk}^k \\
&= \sum_{w \in W_k} c_{0w}^0 + \sum_{v \in V \setminus W_k} \sum_{w \in W_k \mid (v,w) \in A_0} c_{vw}^0 + \sum_{v \in V \setminus W_k \mid (v,k) \in B_k} c_{vk}^k \\
&= \left(|W_k| - \frac{1}{2} \sum_{w \in W_k} \zeta_w + M|W_k| \right) + \frac{1}{2} \sum_{(v,w) \in \delta^-(W_k)} (z_{vw} + z_{wv}) + |V \setminus W_k| M \\
&= nM + |W_k| - \frac{1}{2} \left(\sum_{(v,w) \in \delta^-(W_k)} (z_{vw} + z_{wv}) + 2 \sum_{(v,w) \in A(W_k)} (z_{vw} + z_{wv}) \right) \\
&\quad + \frac{1}{2} \sum_{(v,w) \in \delta^-(W_k)} (z_{vw} + z_{wv}) \\
&= |W_k| - z(A(W_k)) + nM.
\end{aligned}
$$

Therefore, $z(A(W_k)) \leqq |W_k|-1$ holds if and only if $c^k(\delta^-(W_k)) \geqq 1+nM$ holds.

If there is a cut $\delta^-(W_k)$ with $c^k(\delta^-(W_k)) < 1+nM$ we still have to show that $|W_k| \geqq 2$. But this is obvious. Since $k \in W_k$, $|W_k| \geqq 1$. If $W_k = \{k\}$ then $c^k(\delta^-(k)) = 1+nM$. And therefore, $c^k(\delta^-(W_k)) < 1+nM$ implies $|W_k| \geqq 2$. Finally, note that by construction $W_k = V$ is a possible solution. □

Remark 3.9. The separation algorithm for the SEC (3.7) (plus nonnegativity constraints) can be solved by calling n times a max-flow algorithm and is thus solvable in polynomial time.

For the best running time of max-flow algorithms currently known, consult the survey article [2].

Algorithm (3.7) handles a more general situation than we need in the present application. If we assume that the given vector $z \in \mathbb{Q}^A$ satisfies the cardinality constraint (2.3) (1) in addition to the nonnegativity constraints (2.3) (4) then the node set whose related SEC is violated is such that $|W_k| \leqq n-1$. To see this, note that $c^k(\delta^-(V)) = |V| - z(A(V)) + nM = 1+nM$ and so $W_k \subset V$ from Lemma 3.8.

Conversely, if we assume that the given vector $z \in \mathbb{Q}^A$ satisfies the star constraints (2.3) (2) and (3) we can reduce the separation problem to a min-cut problem in an undirected graph (see [1]) and therefore apply the Gomory–Hu algorithm or any other efficient algorithm to compute a minimum capacity cut. We outline this further reduction briefly. Note that if z satisfies (2.3) (2) and (3), then $\zeta_j \leqq 2$ (see (3.3)) for all $j \in V$ and

thus the number M needed in (3.4) can be chosen as zero. This implies that the arc sets B_k introduced in (3.6) and thus the auxiliary graphs D_k, $k \in V$, are not needed. Moreover, we can symmetrize D_0 to a digraph $\tilde{D} = (V_0, \tilde{A})$ with capacities $\tilde{c}_{ij}$ by setting

$$\tilde{A} := A_0 \cup \{(v, 0) \mid v \in V\}, \tag{3.10a}$$

$$\tilde{c}_{ij} := \tilde{c}_{ji} := c_{ij}^0 = c_{ji}^0 = \tfrac{1}{2}(z_{ij} + z_{ji}) \quad \text{for all } (i,j) \in A_0, \tag{3.10b}$$

$$\tilde{c}_{v0} := \tilde{c}_{0v} \quad \text{for all } v \in V. \tag{3.10c}$$

Let $G = (V_0, E)$ be the undirected graph underlying $\tilde{D} = (V_0, \tilde{A})$ with capacities $\hat{c}_{ij}$ defined by

$$E = \{ij \mid (i,j) \in \tilde{A}\}, \tag{3.11a}$$

$$\hat{c}_{ij} := \tilde{c}_{ij} = \tilde{c}_{ji} \quad \text{for all } ij \in E. \tag{3.11b}$$

G has the property that $\hat{c}(\delta(W)) = \tilde{c}(\delta^-(W)) = \tilde{c}(\delta^+(W))$ for any $W \subseteq V$, where $\delta^+(W) = \delta^-(V \setminus W)$. Thus a cut $\delta(W)$ in G with capacity $\hat{c}(\delta(W))$ as small as possible corresponds to a minimum capacity cut $\delta^-(W)$ in $\tilde{D}$ and vice versa.

This construction shows that the separation problem for the SECs—under the assumption that the given point satisfies (2.3) (2)-(4)—can be solved by any algorithm that determines a minimum capacity cut in an undirected graph.

Although the worst case complexity of algorithm (3.7) and the method outlined above are about the same, the latter approach works much better in practice, at least if one uses the method described in [18], as we did.

Let us now turn our attention to the *precedence forcing constraints*, the PFCs,

$$x((j:W)) + x(A(W)) + x((W:i)) \leqq |W| \tag{3.12}$$

$$\text{for all } (i,j) \in R \quad \text{and all} \quad \emptyset \neq W \subseteq V \setminus \{i,j\}.$$

As above, we reduce the separation problem for (3.12) to a series of min-cut problems.

We assume that a point $z \in \mathbb{Q}^A$ satisfying $z_{ij} \geqq 0$ for all $(i,j) \in A$ is given and we want to find an inequality of (3.12) that is violated by z, if one exists. We do this by constructing, for each arc $(i,j) \in R$, a min-cut problem that proves whether or not (3.12) is satisfied for all $\emptyset \neq W \subseteq V \setminus \{i,j\}$.

For every arc $(i,j) \in R$ of the precedence digraph $P = (V, R)$, we introduce a new digraph $D_{ij} = (V_{ij}, A_{ij})$ with capacities d^{ij} as follows.

$$V_{ij} := (V \setminus \{i,j\}) \cup \{v_{ij}\} \quad \text{where } v_{ij} \text{ is a new node,} \tag{3.13a}$$

$$A^z := \{(i,j) \in A \mid z_{ij} > 0\}, \tag{3.13b}$$

$$A_{ij} = \{(k,l) \mid (k,l) \in A^z,\ k, l \notin \{i,j\}\} \cup \{(v_{ij}, l) \mid (j,l) \in A^z,\ l \notin \{i,j\}\} \cup \{(k, v_{ij}) \mid (k,i) \in A^z,\ k \notin \{i,j\}\}, \tag{3.13c}$$

$$d_{kl}^{ij} := z_{kl} \quad \text{for all } (k,l) \in A_{ij} \cap A^z, \tag{3.13d}$$

$$d_{v_{ij}l}^{ij} := z_{jl} \quad \text{for all } (j,l) \in A^z, \tag{3.13e}$$

$$d_{kv_{ij}}^{ij} := z_{ki} \quad \text{for all } (k,i) \in A^z. \tag{3.13f}$$

See Fig. 1 for an illustration. Observe that D_{ij} is obtained from $D^z = (V, A^z)$ by deleting all arcs directed into j, all arcs leaving i, all arcs between i and j, and by identifying the nodes i and j. The capacities are just the values of the (positive) components of z.

The PFCs concerning $(i,j) \in R$ and D_{ij},

$$x((j:W)) + x(A(W)) + x((W:i)) \leqq |W|, \tag{3.14}$$

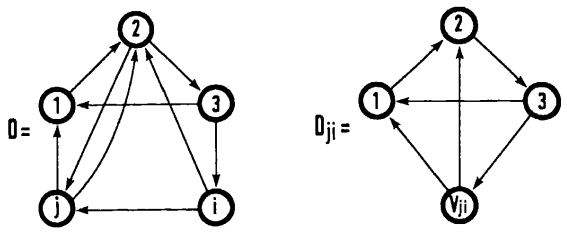

FIG. 1. *Original and shrinking graphs of precedence relationships.*

can be written using this transformation in the form

$$x(A(W\cup\{v_{ij}\}))\leqq|W|=|W\cup\{v_{ij}\}|-1 \tag{3.15}$$

with respect to the digraph D_{ij}. In other words, to check the PFCs concerning $(i,j)\in R$, we have to determine whether the SECs

$$x(A(\tilde W))\leqq|\tilde W|-1 \quad \text{for all } \tilde W\subseteq V_{ij},\ v_{ij}\in\tilde W \text{ and } 2\leqq|\tilde W|\leqq n-1 \tag{3.16}$$

for D_{ij} are satisfied by z. (Recall $n=|V|$ and then $n-1=|V_{ij}|$.) If we can determine a node set $\tilde W\subseteq V_{ij}$ with $v_{ij}\in\tilde W$, $2\leqq|\tilde W|\leqq n-1$, such that $z(A(\tilde W))>|\tilde W|-1$ then, for $W:=\tilde W\backslash\{v_{ij}\}$, $z((j:W))+z(A(W))+z((W:i))>|W|$ obviously holds. If no such $\tilde W$ exists, all inequalities (3.14) concerning $(i,j)\in R$ are satisfied.

By repeating this procedure for all $(i,j)\in R$ we can solve the separation problem for (3.12).

Our task now is to solve the separation problem for (3.15). This can be done by a simplified version of algorithm (3.7). Let $\sigma\equiv v_{ij}$. Normally, we only have to construct in step 1 of (3.7) the auxiliary digraph $D_\sigma=(V_\sigma,A_\sigma)$ with capacities c^σ (associated with the shrunk node σ) from digraph $D_{ij}=(V_{ij},A_{ij})$ with capacities d^{ij}, and perform steps 2 and 3 for this case. If in step c of (3.7) a node set $\tilde W\subseteq V_\sigma$ with $\sigma\in\tilde W$, $0\notin\tilde W$ is identified with $c^\sigma(\delta^-(\tilde W))<1+(n-1)M$ then

$$z(A(\tilde W))>|\tilde W|-1 \tag{3.17}$$

holds and thus the associated precedence forcing constraint is violated. Otherwise, these constraints are satisfied by z. We still have to check whether $|\tilde W|\geqq 2$ holds. But this is obvious since $\sigma\in\tilde W$ and $c^\sigma(\delta^-(\sigma))=1+(n-1)M$. This shows that the separation problem for precedence forcing constraints can be solved in polynomial time for any $z\in\mathbb{Q}^A$ (with $z\geqq 0$). (Note that $\tilde W=V_{ij}$ is allowed in (3.16) and, then, $W=V\backslash(i,j)$ is also allowed.)

If we require that the given $z\in\mathbb{Q}^A$ satisfies (2.3) (2) and (3) in addition—as is the case in our application—we can set the number M equal to zero. This, in fact, simplifies the algorithm a little.

The overall running time of our separation routine for the precedence forcing constraints is at most $O(|R|t)$, where t is the running time for the max-flow algorithm used in step 2 of (3.7). We use the algorithm given in [7].

4. Further inequalities. We note that the sequential ordering problem is closely related to the ATSP and that any inequality valid for the ATSP polytope P_T^n can be brought into a form that is valid for the SOP polytope SOP(n,P) and valid for the set of solutions of (2.3), (2.4), or (2.5).

The classes of valid and facet-defining inequalities for P_T^n (known by 1985) have been surveyed in [16]. In recent years further classes of valid and facet-defining inequalities for P_T^n have been discovered by Balas, Chopra, Fischetti, and Rinaldi, among others. Surveying these achievements here is beyond the scope of this paper. We simply mention those (few) classes of inequalities that we considered in our computations.

The first class consists of the so-called *T_k-inequalities* (most of them facet defining for P_T^n) introduced in [8]. They are defined as follows. Let $k\geqq 2$ and let $W\subseteq V$ be a node set such that $|W|=k$, $w\in W$, and $i,j\in V\backslash W$. Then, the inequality

$$x_{ij}+x_{iw}+x_{wj}+x(A(W))\leqq|W| \tag{4.1}$$

is called a T_k-inequality. Using algorithm (3.7) for the separation problem of the SECs one can easily design a polynomial time algorithm for T_k-inequalities for all k. We did not implement this procedure but used a heuristic to check this type of inequality for $k=2,3,4$.

Other classes of inequalities, facet defining for P_T^n, can be derived by lifting cycle constraints (see [8], [12], and [16]); we use two of these. They are as follows.

For any ordered set of nodes $\{i_1,i_2,\dots,i_k\}\subset V$, $3\leqq k\leqq n-1$,

$$\sum_{\kappa=1}^{k-1}x_{i_\kappa i_{\kappa+1}}+x_{i_k i_1}+2\sum_{\kappa=2}^{k-1}x_{i_\kappa i_1}+\sum_{\kappa=3}^{k-1}\sum_{h=2}^{\kappa-1}x_{i_\kappa i_h}\leqq k-1 \tag{4.2}$$

is called a *D_k^+-inequality* and

$$\sum_{\kappa=1}^{k-1}x_{i_\kappa i_{\kappa+1}}+x_{i_k i_1}+2\sum_{\kappa=3}^{k}x_{i_1 i_\kappa}+\sum_{\kappa=4}^{k}\sum_{h=3}^{\kappa-1}x_{i_\kappa i_h}\leqq k-1 \tag{4.3}$$

is called a *D_k^--inequality*. All D_k^+- and D_k^--inequalities are valid with respect to P_T^n.

We do not know how to solve the separation problem for D_k^+- or D_k^--inequalities in polynomial time unless we fix k and enumerate. This is a ridiculous procedure for large k, but we implemented it for $k=\{3,4\}$.

There are two more liftings of four-cycle inequalities that are facet defining for P_T^n and that we checked by enumeration. These inequalities are of the following types. Again let i_1, i_2, i_3, i_4 be four nodes of V; then the inequalities

$$\sum_{\kappa=1}^{3}x_{i_\kappa i_{\kappa+1}}+x_{i_4 i_1}+2x_{i_2 i_1}+x_{i_2 i_4}+x_{i_3 i_1}+x_{i_4 i_3}\leqq 3, \tag{4.4}$$

$$\sum_{\kappa=1}^{3}x_{i_\kappa i_{\kappa+1}}+x_{i_4 i_1}+2x_{i_1 i_3}+2x_{i_3 i_1}\leqq 3 \tag{4.5}$$

are valid for SOP(n,P). Clearly, they are only useful if all arcs used in (4.4) or (4.5) occur in A.

We are aware of the fact that there are more valid and facet-defining inequalities for P_T^n that might be of interest for solving the sequential ordering problem. For instance, the class of *two-matching* inequalities (in their asymmetric version) could also be considered, in particular, since a polynomial time separation routine is available that is a straightforward adaptation of the method of Padberg and Rao [17] designed for the symmetric case. Moreover, *comb* and *clique tree* inequalities could be used in their asymmetric form since they turned out to be very useful for solving the symmetric TSP in practice (see [9], [19], and [20]). In our case, however, the scope was more limited towards finding good lower bounds for not too large problem instances and,

due to the requirements from practice, no attempt was made to solve the given problems to optimality. Clearly, if one intends to attack truly large scale SOP instances all these classes of inequalities have to be considered.

We should also mention that the idea to separate by enumeration the "small inequalities" listed above was motivated by studying fractional solutions that could not be cut off by SECs or PFCs. The "small inequalities" frequently did the job.

Let us remark, moreover, that most of the inequalities for P_T^n can be extended to take care of precedences in the same way as the SECs were extended to PFCs. To give an example, take the inequality (facet defining for P_T^n)

$$x_{i_1 i_2} + x_{i_2 i_3} + x_{i_3 i_1} + 2x_{i_2 i_1} \leqq 2. \tag{4.6}$$

Assume that (i, j) belongs to R and that the node v_{ij} obtained by identifying nodes i and j (see (3.13)) is the node i_1; then the inequality

$$x_{ji_2} + x_{i_2 i_3} + x_{i_3 l} + 2x_{i_2 l} \leqq 2 \tag{4.7}$$

is valid for SOP(n, P). This type of SOP extension can be made in various ways. We have implemented separation routines for a few of them but do not want to discuss the simple but rather technical details.

5. Preprocessing. A (usually important) part of a cutting plane procedure consists of analyzing the given problem instance in order to discover some structure that helps to decompose the instance, to reduce its size, or to tighten the IP-formulation by turning some inequalities into equations, fixing certain variables, etc.

We do not want to elaborate on all preprocessing routines that we have implemented; we simply list a few of the straightforward cases. We concentrate here on the IP-formulation (2.5) of the SOP. Suppose the complete digraph $D_n = (V, A_n)$ with cost c_{ij} for all $(i, j) \in A$, and the acyclic and transitively closed precedence digraph $P = (V, R)$ are given. In a first step we determine the node sets V^- and V^+ as follows:

$$V^- = \{v \in V \mid \exists (i, v) \in R,\ i \neq v\}, \tag{5.1a}$$

$$V^+ = \{v \in V \mid \exists (v, j) \in R,\ j \neq v\}, \tag{5.1b}$$

i.e., V^- is the set of nodes that have predecessors in P, and V^+ is the set of nodes that have successors in P. It is obvious that the inequalities (2) and (3) of (2.3) can be transformed into

$$(2') \quad x(\delta^-(j)) = 1 \quad \text{for all } j \in V^-,$$

$$(2) \quad x(\delta^-(j)) \leqq 1 \quad \text{for all } j \in V \setminus V^-,$$

$$(3') \quad x(\delta^+(j)) = 1 \quad \text{for all } j \in V^+,$$

$$(3) \quad x(\delta^+(j)) \leqq 1 \quad \text{for all } j \in V \setminus V^+.$$

Since we drop all variables corresponding to arcs in $A_n \setminus A$ it may happen that by logical implication some of the inequalities (2) or (3) can also be turned into equations. This type of analysis is made not only in the preprocessing phase but also in all later steps when certain variables can be fixed to zero or 1 due to reduced cost criteria. Again, we do not want to discuss the obvious and well-known details of this technique.

Another preprocessing step that is based on an analysis of the precedence digraph P and the cost values c_{ij} turned out to be quite useful in solving some of our cases, due to their special cost matrix structure. It sometimes happens that, for two nodes i and j that are unrelated for the given precedences, an "artificial" precedence, say (i, j), can be introduced (by analyzing the cost matrix) in such a way that the optimum value

of the SOP before and after introducing the relation (i, j) is the same. In such a case we can repeat the other preprocessing steps such as variable fixing and inequality tightening, and start the whole process anew.

To show how an "artificial" precedence can be created we consider a small example on seven nodes. The cost matrix is shown in Table 1. The precedence digraph $P = (V, R)$ is given by $R = \{(1, j) \mid j = 4, 5, 6, 7\} \cup \{(i, 5) \mid i = 1, 4, 6, 7\}$. Nodes 2 and 3 are not related to any other node.

Consider the two unrelated nodes 2 and 7. We observe that $c_{27} = c_{72} = 0$. Moreover, $c_{2k} = c_{7k}$ and $c_{k7} = c_{k2}$ hold for all $k \in V \setminus \{2, 7\}$. In addition, we can observe that $c_{uv} \leqq c_{u2} + c_{2v}$ for all $u, v \in V \setminus \{2, 7\}$, $u \neq v$. Since node 2 is not related to any other node in P we can either add the arc $(2, 7)$ or the arc $(7, 2)$ to P, and we can also set $x_{27} = 1$ or $x_{72} = 1$, without changing the objective function (cost) value of the optimal solution.

It is easy to see how to generalize this observation. If there are two nodes $i, j \in V$ such that

$$(i, j), (j, i) \notin R, \tag{5.2a}$$

$$c_{ij} = c_{ji} = 0, \tag{5.2b}$$

$$c_{ik} = c_{jk} \quad \text{and} \quad c_{ki} = c_{kj} \quad \text{for all } k \in V \setminus \{i, j\}, \tag{5.2c}$$

$$c_{uv} \leqq c_{uj} + c_{jv} \quad \text{for all } u, v \in V \setminus \{i, j\}, \quad u \neq v, \tag{5.2d}$$

$$j \notin (V^- \cup V^+) \quad \text{(i.e., } j \text{ has neither a predecessor nor a successor in } P), \tag{5.2e}$$

then either (i, j) or (j, i) can be added to R, and either x_{ij} or x_{ji} can be set to 1, such that at least one optimum solution of the original SOP instance is still optimum for the new case.

It turned out that this "precedence addition rule" helped in some cases to substantially reduce the problem size and to increase the lower bound from the LP relaxation.

6. Outline of the implementations. We have made three new implementations of cutting plane algorithms that compute lower bounds for the SOP. Two algorithms use model (2.5) and one uses model (2.4). Moreover, we compared this with the algorithm for the LP relaxation of model (2.3) described in [4].

Implementation A is based on model (2.4) and implementation B is based on model (2.5). Both were coded in FORTRAN, used Marsten's simplex-based LP solver XMP (see [15]), and were implemented and executed on a SIEMENS PC MX-2 (a 0.7 MIPS personal computer with UNIX operating system).

TABLE 1
Cost coefficients.

i \ j	1	2	3	4	5	6	7
1	—	1.00	2.00	0.75	0.00	3.00	1.00
2	4.00	—	5.00	3.25	4.00	6.00	0.00
3	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	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	—

Implementation C is based on model (2.5). It was coded in PL/I version 1.5, used the algorithmic tools of the LP-solver MPSX (see [14]), and was implemented and executed on an IBM 4381 (a 7.7 MIPS computer with VM/CMS operating system).

Implementations A and B were meant to determine which of the two models (2.4) and (2.5) are superior from a computational point of view. We also wanted to see whether or not SOP instances of the size coming up in practice (up to about 100 nodes and 280 precedence relationships) can be solved in reasonable time on a PC.

We now briefly outline the basics of our cutting plane approach. We concentrate mainly on the codes B and C that solve the LP relaxations of model (2.5).

The algorithm receives an $n \times n$ cost matrix and an acyclic digraph of precedences as input. We may assume that all cost coefficients are integral (for expository purposes). In a first step we compute the transitive hull of this digraph to obtain the initial precedence subdigraph. In a second step we try to add precedence relations by analyzing the cost matrix as described in § 5. If we add a precedence, we recompute the transitive hull and repeat until no further precedence can be added. We denote the final precedence subdigraph by $P=(V, R)$.

Then we compute the arc set $A=A_n\setminus(\bar{R}\cup\vec{R})$ (see (2.1)) and we try to find out whether further arcs can be deleted from A (or fixed) by analyzing logical implications.

Now we set up the initial LP consisting of (2.3) (1)-(4), taking care that (as outlined in § 5) some of the inequalities can be turned into equations.

To solve model (2.4) we also set up (7), (8), (10), and (11). In this case we project away half of the variables y_{ij}'s using (8) and we fix some of the variables y_{ij}'s appropriately according to the previous fixing of variables x_{ij}'s.

We now run the heuristic described in [4] and [5] to find a "good" feasible Hamiltonian path. Let λ^H denote its cost. We use it to set up an initial basis for the LP-solver.

We solve the present LP and obtain an optimum solution z with value λ_{LP}. If z is the incidence vector of a feasible Hamiltonian path we are done. We are also done if $\lambda^H-\lambda_{LP}<1$. In this case the heuristically found feasible Hamiltonian path is optimal.

Otherwise we enter the separation process. We first check whether z satisfies the SECs and then the PFCs using the separation algorithms described in § 3. We add all inequalities found this way to the current LP. If z satisfies all SECs and all PFCs then we call the separation algorithms for the further inequalities mentioned in § 4. Again we add all inequalities found to the current LP.

If the second stage of separation routines fails, we finish the cutting plane algorithm reporting the lower bound λ_{LP}.

Otherwise we continue, but before resolving the augmented LP, we do the well-known reduced cost fixing of variables. If some of the variables can be fixed, we determine the logical implications in order to fix further variables. Moreover, we call the preprocessing routines to tighten the current LP further. In addition, we delete redundant constraints. After these preparations we call the LP solver using the modified LP and the old (dually feasible) basis.

When solving model (2.4), we additionally check the triangle inequalities (9) by enumeration and add all inequalities found to the present LP.

This finishes the outline of our implementations. There are many technical details that we think are important, but it is impossible to report all of them here. The codes B and C, although following the same ideas, do not always produce the same value λ_{LP}, since they were written by different people, and some differences in the order of performing certain steps, setting tolerances, etc., caused variations in the running times and the LP values. In particular, implementation C does not use the T_k-inequalities (4.1), nor the mechanism for generating "artificial" precedences based on the cost matrix structure (see § 5). On the other hand, the D_k^+- and D_k^--inequalities (4.2) and (4.3) as well as the other further inequalities were only used at selected LP problems, where a given analysis of the point z may suggest a potential violation of the constraints; of course, there is no guarantee that all violations were investigated. Additionally, implementation C temporarily declares "neutral" certain currently nonactive inequalities based on counting the number of previous consecutive LPs where they have been nonactive.

For illustrative purposes let us consider the case described in § 5; see also Table 1. The heuristic gives the feasible solution $1\to4\to2\to7\to6\to5\to3$. The total cost is $\lambda^H=2125$. The optimal value of the LP model (2.3) (1)-(4) is $\lambda_{LP}=1800$; it gives the solution $1\to4\to6\to5\to3$ and $2\to7\to2$. By using our separation algorithm for model (2.5) but without considering the cost matrix structure, the optimal solution of the augmented LP is $\lambda_{LP}=2075$ (then the gap is 2.40 percent). By exploiting the cost structure as described in § 5 we force the precedence $(2,7)\in R$ and then update $A:=A\setminus\{(7,2),(5,2),(2,5)\}$. It turns out that the optimal value of the new LP is precisely $\lambda_{LP}=2125$.

7. Computational results. We now report some computational experiences with the three implementations of the cutting plane algorithms outlined in § 6 and compare these with the heuristic described in [4] and [5] and the lower bounding algorithm described in [4].

The report covers 16 instances of the SOP where the number n of tasks ranges from 7 to 98 and the number $|R|$ of precedence relationships from 0 to 283. Six of these cases are real-life and came up in a scheduling system for manufacturing. Four further cases ($P1$, $P1A$, $P4$, and $P9$) were created (artificially) to test certain aspects of the cut generation, mainly the performance of exploiting the cost structure. The remaining six cases are obtained from the real-life cases by dropping all precedence relationships. So these are, in fact, "pure" Hamiltonian path problems.

The artificial cases were constructed as follows. Cases $P4$ and $P9$ are created by replicating case $P1$ $\rho=2$ and 14 times, respectively. (Case $P1$ is described in § 5.) Node i in $P1$ has the counterparts $nj=i+7(j-1)$ for $i=1,2,\dots,7$ in cases $P4$ and $P9$, $j=1,2$ in $P4$ and $j=1,2,\dots,14$ in $P9$. The sets of nodes $\{i=1,2,\dots,7\}$ and $\{nj=8,9,\dots,14\}$ in case $P4$ have the same internal precedence relationships as the set of nodes in $P1$ have; on the other hand, none of the nodes from one set has precedence relationships with the nodes from the other set. The cost matrix has the following structure for $P4$: $c_{p,q+7}=c_{p+7,q}=c_{p+7,q+7}=c_{p,q}$, $c_{p,q}$ for $p,q=1,2,\dots,7$, $p\neq q$ as in $P1$, and the other elements are zero. (A similar construction is used for $P9$.) The optimal solution is $N1\to N4\to N2\to N7\to N6\to N5\to N3$ with $\lambda^*=2125$, where, e.g., $N6$ denotes any sequencing of the node set $\{6,13,\dots,k\}$ for $k=6+7(\rho-1)$. Note that, naturally, the optimal value of the LP relaxation (2.3) (1)-(4) is zero for $P4$ and $P9$. By exploiting the cost matrix structure as in § 5 (see (5.2)), implementation B gets the optimal solution without adding any further cut. Additionally, we also analyze the performance of our separation algorithm when the cost matrix structure is not exploited.

Table 2 reports some results on the performance of our algorithm. It gives information about the objective function value and the gap between the best-known upper bound (frequently, the optimal solution value) and the lower bounds obtained by our implementations. The headings are as follows. H refers to the heuristic described in [4] and [5]. E refers to the algorithm described in [4] for obtaining a lower bound of the optimal solution value λ^* in model (2.3). Finally, A, B, and C refer to our three

TABLE 2
Performance of our cutting separation implementations.

Case	n	$\|R\|$	Objective function value					Gap in objective function			
			H	E	A	B	C	E	A	B	C
P1	7	7	2125	1950	2125	2125	2075	8.97	0.00	0.00	2.40
P1A	7	0	550	450	550	550	550	22.22	0.00	0.00	0.00
P2	11	5	2075	2021	2075	2075	2075	2.70	0.00	0.00	0.00
P2A	11	0	1866	1763	1866	1866	1843	5.80	0.00	0.00	1.25
P3	12	11	1675	1417	1598	1597	1535	18.20	4.82	4.88	9.12
P3A	12	0	1472	1386	1472	1472	1459	6.20	0.00	0.00	0.89
P4	14	14	2125	1525	2125	2125	2075	39.34	0.00	0.00	2.40
P5	25	11	1684	1518	1588	1577	1584	10.90	6.05	6.79	6.31
P5A	25	0	1145	1041	1134	1141	1118	10.00	0.97	0.35	2.42
P6	47	32	1288	1199	1219	1218	1219	7.40	5.66	5.75	5.66
P6A	47	0	915	856	872	872	871	6.09	4.93	4.93	5.05
P7	63	233	63	63	62	62	63	0.00	1.61	1.61	0.00
P7A	63	0	45	45	45	45	45	0.00	0.00	0.00	0.00
P8	78	283	18480	18205	18205	18205	18205	1.51	1.51	1.51	1.51
P8A	78	0	1845	1410	1305	1845	1712	30.85	41.37	0.00	7.76
P9	98	98	2125	1525	2125	2125	2075	39.34	0.00	0.00	2.40

implementations A, B, and C (see § 6). The first part of Table 2 reports the cost, say λ^H, of the heuristic solution (except for *P8A*; see below) as well as the lower bound, say λ_a, obtained by implementation a for $a = $ A, B, C, and E. It is worth noting that the heuristic gives 2325 as the cost value for *P8A*, but one of our implementations found the (optimal) value 1845. The gap as reported in Table 2 is $100(\lambda^H - \lambda_a)/\lambda_a$. Note that frequently the gap is zero (i.e., implementations A, B, and C prove the optimality of the heuristic solution). We should mention that the gap for *P1*, *P4*, and *P9* is 2.40 percent when the cost matrix structure is not exploited (as with implementation C) and it is zero when it is exploited.

TABLE 3
CPU *time of our cutting separation implementations.*

Case	n	$\|R\|$	H (sec.)	E (sec.)	A (min.)	B (min.)	C (sec.)
P1	7	7	0.08	0.10	0.14	0.10	0.05
P1A	7	0	0.13	0.24	0.18	0.10	0.05
P2	11	5	0.26	0.55	0.17	0.20	0.04
P2A	11	0	0.22	0.55	1.05	0.15	0.09
P3	12	11	0.16	0.36	9.59	1.27	0.89
P3A	12	0	0.25	0.56	1.06	0.18	0.21
P4	14	14	0.31	1.10	0.17	0.10	0.71
P5	25	11	1.17	1.19	23.34	0.43	0.55
P5A	25	0	0.35	1.28	13.25	0.46	0.45
P6	47	32	3.05	4.78	87.39	4.51	1.14
P6A	47	0	1.98	3.85	100.10	1.20	1.08
P7	63	233	4.35	3.38	340.43	27.27	5.63
P7A	63	0	0.06	3.47	109.37	1.42	4.08
P8	78	283	27.62	12.93	443.13	153.01	12.05
P8A	78	0	8.93	6.94	250.52	9.36	18.41
P9	98	98	28.47	25.01	0.23	0.20	6.20

We should remark at this point that implementation A (using the largest number of variables) was not able to finish all runs due to space limitations on the PC. (We could not store all inequalities found.) In this case we report in Table 2 the lower bound obtained before termination. (Further cutting plane steps might have led to better lower bounds.)

Table 3 reports the CPU time required by the implementations. Implementations H, E, and C were run on an IBM 4381 and the time is given in seconds. Implementations A and B were run on a SIEMENS PC MX-2 and the time is given in minutes. All times reported include input-output operations. Note that the PC-implementation B solves the cases in less than $2\frac{1}{2}$ CPU hours. The mainframe version does this in a few seconds.

Tables 4–6 report the dimensions of the instances and number of cuts that have been generated by each of the three implementations. The headings are as follows. F01 indicates the number of variables x_{ij} that are (permanently) fixed by reduced cost fixing and logical implications. (Note that $|A|$ gives the set of variables x_{ij}'s in the model and, then, $|A| -$ F01 is the number of x_{ij}'s in the last LP problem.) NAP is the number of constraints (2.3) (1)–(3) (i.e., number of constraints in the initial LP relaxation) in implementations B and C; NAP is the number of constraints (2.4) (1)–(3), (7), and (8) in implementation A; NLP is the number of cutting plane separation steps (i.e., number of LP problems); NSEC is the number of subtour elimination constraints (2.3) (5) that have been generated; NYSC is the number of y-related constraints (2.4) (9) that have been generated in implementation A; NPFC is the number of precedence forcing constraints (2.5) (12) that have been generated in implementations B and C; NLC is the number of further cuts generated from the class of inequalities described in § 4; NC is the total number of cuts that have been generated. One can observe that the total number of constraints in any LP is rather small. By comparing NLP and NC we can see the average number of cuts that are appended to the LP model at each iteration.

TABLE 4
Problem dimensions and cut generation. Implementation A.

Case	n	$\|R\|$	$\bar{R}$	$\|A\|$	F01	NAP	NLP	NSEC	NYSC	NLC	NC
P1	7	7	1	35	10	39	2	1	10	0	11
P1A	7	0	0	63	26	57	1	0	0	0	0
P2	11	5	2	153	80	123	4	2	56	6	64
P2A	11	0	0	165	89	133	3	2	49	0	51
P3	12	11	4	172	53	135	7	4	64	29	97
P3A	12	0	0	198	98	157	2	2	41	0	43
P4	14	14	2	35	10	39	2	1	10	0	11
P5	25	11	2	876	496	629	8	4	667	11	682
P5A	25	0	0	900	0	651	6	3	434	0	437
P6	47	32	22	3157	1890	2199	15	2	1502	0	1504
P6A	47	0	0	3243	1949	2257	5	5	1800	0	1805
P7	63	233	138	5255	2134	3567	4	10	1800	13	1823
P7A	63	0	0	5859	2957	4033	1	0	0	0	0
P8	78	283	206	8237	2079	5597	3	12	600	8	612
P8A	78	0	0	9009	0	6163	1	1	200	8	209
P9	98	98	14	35	12	39	2	1	10	0	11

TABLE 5
Problem dimensions and cut generation. Implementation B.

Case	n	$\|R\|$	$\bar{R}$	$\|A\|$	F01	NAP	NLP	NSEC	NPFC	NLC	NC
P1	7	7	1	23	10	15	2	1	1	0	2
P1A	7	0	0	42	28	15	2	1	0	0	1
P2	11	5	2	103	80	23	4	4	9	0	13
P2A	11	0	0	110	89	23	4	3	0	0	3
P3	12	11	4	117	53	25	12	9	6	22	37
P3A	12	0	0	132	98	25	3	10	0	0	10
P4	14	14	2	166	10	29	2	1	1	0	2
P5	25	11	2	587	498	51	3	5	12	0	17
P5A	25	0	0	600	550	51	4	5	0	0	5
P6	47	32	22	2108	1870	95	4	7	7	0	14
P6A	47	0	0	2162	1956	95	4	8	0	0	8
P7	63	233	138	3535	2802	127	7	8	63	65	136
P7A	63	0	0	3906	2957	127	1	0	0	0	0
P8	78	283	206	5517	2079	157	15	11	65	66	142
P8A	78	0	0	6006	1547	157	18	31	0	16	47
P9	98	98	14	9492	—	197	2	1	1	0	2

TABLE 6
Problem dimensions and cut generation. Implementation C.

Case	n	$\|R\|$	$\bar{R}$	$\|A\|$	F01	NAP	NLP	NSEC	NPFC	NLC	NC
P1	7	7	1	23	14	15	2	2	2	3	7
P1A	7	0	0	42	31	15	2	2	0	0	2
P2	11	5	2	103	78	23	5	4	12	6	22
P2A	11	0	0	110	83	23	5	3	0	0	3
P3	12	11	4	117	25	25	11	9	6	12	27
P3A	12	0	0	132	74	25	3	9	0	0	9
P4	14	14	2	166	20	29	2	3	6	2	11
P5	25	11	2	587	469	51	3	5	9	0	14
P5A	25	0	0	600	505	51	4	4	0	0	4
P6	47	32	22	2108	1842	95	7	7	6	0	13
P6A	47	0	0	2162	1925	95	4	7	0	0	7
P7	63	233	138	3535	3227	127	7	9	0	0	9
P7A	63	0	0	3906	2603	127	3	4	0	0	4
P8	78	283	206	5517	4748	147	17	15	72	16	103
P8A	78	0	0	6006	2079	157	9	28	0	13	41
P9	98	98	14	9492	4723	197	2	3	11	3	17

Table 7 reports the gap reduction on the objective function value obtained by our implementations. The headings are as follows. H is the best known upper bound of λ^*. ALB is the objective function value of the LP relaxation (2.3) (1)-(3). A-GAP = (H − ALB)/ALB percent. KLB is our best-known lower bound on λ^* (i.e., the objective function value of the last LP). K-GAP = (H − KLB)/KLB percent and RK = (KLB − ALB)/(H − ALB) percent.

The first analysis that we can draw from the results shown in Table 7 is the observance of a big discrepancy in the value of A-GAP between the results reported in the literature for randomly generated cases and our experience with real-life cases. It is reported for ATSP cases that the value of ALB was found on the average to be 99.5 percent of the optimal value. We have obtained the optimal value in more than

TABLE 7
Gap reduction on the objective function value.

Case	n	$\|R\|$	H	ALB	A-GAP	KLB	K-GAP	RK
P1	7	7	2125	1800	18.06	2125	0.00	100.00
P1A	7	0	550	225	100.00	550	0.00	100.00
P2	11	5	2075	1946	6.63	2075	0.00	100.00
P2A	11	0	1866	1763	5.84	1866	0.00	100.00
P3	12	11	1675	1293	29.54	1598	4.88	79.58
P3A	12	0	1472	1240	18.71	1472	0.00	100.00
P4	14	14	2125	0	*	2125	0.00	100.00
P5	25	11	1684	1518	10.94	1588	6.05	42.16
P5A	25	0	1145	1041	9.99	1141	0.35	96.15
P6	47	32	1288	1199	7.42	1219	5.66	22.47
P6A	47	0	915	856	6.89	872	4.93	27.11
P7	63	233	63	62	1.61	63	0.00	100.00
P7A	63	0	45	45	0.00	45	0.00	*
P8	78	283	18480	18204	1.52	18205	1.51	0.36
P8A	78	0	1845	1305	41.37	1845	0.00	100.00
P9	98	98	2125	0	*	2175	0.00	100.00

50 percent of the cases and we have at hand the lower bound KLB for the other subset; we have to report a big difference between H and ALB and even KLB and ALB.

We should point out the effectiveness of the separation algorithm for identifying subtour elimination constraints that are violated by the current LP solution. See column RK in Table 7 for the ATSP cases (i.e., cases with $|R| = 0$). It gives the gap reduction obtained by appending violated SECs to the current LP model. On the other hand, we may observe the performance of the preprocessing procedure based on (2.1) ($|A_n \backslash A|$ variables x_{ij}'s are fixed to zero) for tightening the lower bound ALB for the cases with precedence relationships (i.e., cases with $|R| > 0$). Note also how effective the reduced cost fixing can be whenever ALB and H are close enough. Finally, see that ALB is zero for *P*4 and *P*9 in implementation C (i.e., the cost matrix structure is not exploited).

The column headed KLB in Table 7 gives our tightest lower bound on the SOP optimal solution. It is the optimal value of the LP relaxation (2.3) (1)-(4) enlarged by appending the cuts that our separation algorithm identifies as violated cuts. By comparing the columns headed A-GAP and K-GAP and, in particular, analyzing the column headed RK, we can see the effectiveness of appending violated cuts. Notice that the optimality of the solution provided by the heuristic has been proved for 9 out of 16 cases. On the other hand, the largest gap is only 6.05 percent. Branch-and-bound has not been used, since our only objective was to create (hopefully) good lower bounds for the heuristic given in [4] and [5].

8. Conclusions. In this work we have presented two new 0-1 models for the sequential ordering problem. Both are stronger than the model introduced in [4]. We have also introduced polynomial time separation algorithms for subtour elimination constraints and precedence forcing constraints. We have outlined the LP framework of three implementations for tightening the lower bound of the optimal solution and reported our computational results. More theoretical work is required mainly for identifying (in reasonable time) violated further inequalities mentioned in § 5. In any case, our computational experience indicates that this LP-based approach is a quite promising way to analyze the quality of a feasible solution and eventually to obtain an optimal one.

Acknowledgment. We appreciate Manfred Padberg's and Giovanni Rinaldi's comments.

REFERENCES

[1] N. ASCHEUER, L. F. ESCUDERO, M. GRÖTSCHEL, AND M. STOER, *On identifying in polynomial time violated subtour elimination and precedence forcing constraints for the sequential ordering problem*, in Integer Programming and Combinatorial Optimization, R. Kannan and W. R. Pulleyblank, eds., University of Waterloo, Waterloo, Ontario, Canada, 1990, pp. 19–28.

[2] R. K. AHUJA, T. L. MAGNANTI, AND J. B. ORLIN, *Network flows*, in Optimization, G. L. Nemhauser, A. H. G. Rinnooy Kan, and M. J. Todd, eds., North-Holland, Amsterdam, 1989, pp. 211–369.

[3] H. CROWDER AND M. PADBERG, *Solving large-scale symmetric traveling salesman problems to optimality*, Management Sci., 26 (1980), pp. 495–509.

[4] L. F. ESCUDERO, *An inexact algorithm for the sequential ordering problem*, European J. Oper. Res., 37 (1988), pp. 236–253.

[5] ———, *On the implementation of an algorithm for improving a solution to the sequential ordering problem*, Trabajos de Investigación-Operativa, 3 (1988), pp. 117–140.

[6] ———, *A production planning problem in* FMS, Ann. Oper. Res., 17 (1989), pp. 69–104.

[7] A. V. GOLDBERG AND R. E. TARJAN, *A new approach to the maximum flow problem*, Assoc. Comput. Mach., 35 (1988), pp. 921–940.

[8] M. GRÖTSCHEL, *Polyedrische Charakterisierungen kombinatorischer Optimierungsprobleme*, Hain, Meisenheim am Glan, Germany, 1977.

[9] M. GRÖTSCHEL AND O. HOLLAND, *Solution of large-scale symmetric travelling salesman problems*, Math. Programming, 51 (1991), pp. 191–202.

[10] M. GRÖTSCHEL, M. JÜNGER, AND G. REINELT, *A cutting plane algorithm for the linear ordering problem*, Oper. Res., 34 (1984), pp. 1195–1220.

[11] M. GRÖTSCHEL, L. LOVASZ, AND A. SCHRIJVER, *Geometric Algorithms and Combinatorial Optimization*, Springer-Verlag, Berlin, 1988.

[12] M. GRÖTSCHEL AND M. PADBERG, *Lineare Charakterisierungen von Travelling Salesman Problemen*, Z. Oper. Res., 21 (1977), pp. 33–64.

[13] K. HOFFMAN AND M. PADBERG, LP-*based combinatorial problem solving*, Ann. Oper. Res., 5 (1986), pp. 145–194.

[14] IBM, *Mathematical Programming System Extended* (MPSX/370) *Version* 2, Program reference manual, SH19-6553, 1988.

[15] R. E. MARSTEN, *The design of the* XMP *linear programming library*, ACM Trans. Math. Programming Software, 7 (1981), pp. 481–497.

[16] M. PADBERG AND M. GRÖTSCHEL, *Polyhedral computations*, in The Traveling Salesman Problem, A Guided Tour of Combinatorial Optimization, E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy-Kan, and D. B. Shmoys, eds., John Wiley, New York, 1985, pp. 251–360.

[17] M. PADBERG AND M. R. RAO, *Odd minimum cut-sets and b-matchings*, Math. Oper. Res., 7 (1982), pp. 67–80.

[18] M. PADBERG AND G. RINALDI, *An efficient algorithm for the minimum capacity cut problem*, Math. Programming, 47 (1990), pp. 19–36.

[19] ———, *Facet identification for the symmetric travelling salesman polytope*, Math. Programming, 47 (1990), pp. 219–258.

[20] ———, *Optimization of a* 532*-city symmetric traveling salesman problem*, Oper. Res. Lett., 6 (1987), pp. 1–7.

[21] ———, *A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems*, SIAM Rev., 33 (1991), pp. 60–100.

[22] W. PULLEYBLANK AND M. FIALA-TIMLIN, *Precedence constrained routing*, ORSA/TIMS Joint National Meeting, New York, 1989.

SIAM J. OPTIMIZATION
Vol. 3, No. 1, pp. 43–59, February 1993

003

ERROR BOUND AND REDUCED-GRADIENT PROJECTION ALGORITHMS FOR CONVEX MINIMIZATION OVER A POLYHEDRAL SET*

ZHI-QUAN LUO† AND PAUL TSENG‡

Abstract. Consider the problem of minimizing, over a polyhedral set, the composition of an affine mapping with a strongly convex differentiable function. The polyhedral set is expressed as the intersection of an affine set with a (simpler) polyhedral set and a new local error bound for this problem, based on projecting the reduced gradient associated with the affine set onto the simpler polyhedral set, is studied. A class of reduced-gradient projection algorithms for solving the case where the simpler polyhedral set is a box is proposed and this bound is used to show that algorithms in this class attain a linear rate of convergence. Included in this class are the gradient projection algorithm of Goldstein and Levitin and Poljak, and an algorithm of Bertsekas. A new algorithm in this class, reminiscent of active set algorithms, is also proposed. Some of the results presented here extend to problems where the objective function is extended real valued and to variational inequality problems.

Key words. local error bound, convex minimization, linear convergence, reduced-gradient projection algorithms

AMS(MOS) subject classifications. 49, 90

1. Introduction.

We consider the convex program

$$\begin{aligned} &\text{minimize} && f(x) \\ &\text{subject to} && x \in \mathcal{X}, \end{aligned} \tag{1.1}$$

where $\mathcal{X}$ is a polyhedral set in the n-dimensional Euclidean space $\Re^n$ and f is a real-valued function defined on $\Re^n$. We assume that f is of the special form

$$f(x) = g(Ex) + \langle q, x \rangle, \tag{1.2}$$

where E is some $m \times n$ matrix, q is some vector in $\Re^n$, and g is a continuously differentiable function in $\Re^m$ with ∇g Lipschitz continuous and strongly monotone in the sense that there exist positive scalars $\rho > 0$ and $\sigma > 0$ such that

$$\|\nabla g(z) - \nabla g(w)\| \le \rho \|z - w\| \quad \forall z, \quad \forall w, \tag{1.3}$$

and

$$\langle \nabla g(z) - \nabla g(w), z - w \rangle \ge \sigma \|z - w\|^2 \quad \forall z, \quad \forall w. \tag{1.4}$$

We also assume that the optimal solution set for (1.1), denoted by $\mathcal{X}^*$, is nonempty and denote by v^* the value of f on $\mathcal{X}^*$. In our notation, all vectors are column vectors, superscript T denotes matrix transpose, $\langle \cdot, \cdot \rangle$ denotes the usual Euclidean inner product, and $\| \cdot \|$ denotes the Euclidean norm induced by $\langle \cdot, \cdot \rangle$.

* Received by the editors January 3, 1991; accepted for publication (in revised form) November 18, 1991.

† Communications Research Laboratory, Room 225, McMaster University, Hamilton, Ontario, L8S 4K1, Canada (luozq@sscvax.cis.mcmaster.ca). The research of this author is supported by Natural Sciences and Engineering Research Council of Canada grant OPG0090391.

‡ Department of Mathematics, GN-50, University of Washington, Seattle, Washington 98195 (tseng@math.washington.edu).