A cutting plane algorithm for the single machine scheduling problem with release times

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Mathematics and Computing Science

COSOR-Memorandum 91-05

A Cutting Plane Algorithm for the Single Machine Scheduling Problem with Release Times

by

G.L. Nemhauser M.W.P. Savelsbergh

February 1991

A Cutting Plane Algorithm for the Single Machine Scheduling Problem with Release Times

G.L. Nemhauser [1,2]
Georgia Institute of Technology, Atlanta

M.W.P. Savelsbergh [1,3]
Eindhoven University of Technology

Abstract

We propose a mixed integer programming formulation for the single machine scheduling problem with release times and the objective of minimizing the weighted sum of the start times. The basic formulation involves start time and sequence determining variables, and lower bounds on the start times. Its linear programming relaxation solves problems in which all release times are equal. For the general problem, good lower bounds are obtained by adding additional valid inequalities that are violated by the solution to the linear programming relaxation. We report computational results and suggest some modifications based on including additional variables that are likely to give even better results.

[1]Supported by NATO Collaborative Research Grant No. 901057
[2]Supported by NSF Research Grant No. ISI-8761183
[3]Supported by the Netherlands Organization for Scientific Research through NATO Science Fellowship Grant No. N62-316.89

A Cutting Plane Algorithm for the Single Machine Scheduling Problem with Release Times

G.L. Nemhauser
Georgia Institute of Technology, Atlanta

M.W.P. Savelsbergh
Eindhoven University of Technology

Abstract

We propose a mixed integer programming formulation for the single machine scheduling problem with release times and the objective of minimizing the weighted sum of the start times. The basic formulation involves start time and sequence determining variables, and lower bounds on the start times. Its linear programming relaxation solves problems in which all release times are equal. For the general problem, good lower bounds are obtained by adding additional valid inequalities that are violated by the solution to the linear programming relaxation. We report computational results and suggest some modifications based on including additional variables that are likely to give even better results.

1 Introduction

Recently developed polyhedral methods have yielded substantial progress in solving many important NP-hard combinatorial optimization problems. Some well-known examples are the traveling salesman problem [Grötschel and Padberg 1985a, Grötschel and Padberg 1985b], the acyclic subgraph problem [Jünger 1985], and large scale 0-1 integer programming problems [Crowder, Johnson and Padberg 1983]. See Hoffman and Padberg [1985] and Nemhauser and Wolsey [1988] for general descriptions of this approach.

However, for mixed-integer problems, in particular machine scheduling, polyhedral methods have not been nearly so successful. Investigation and development of polyhedral methods for machine scheduling problems is important because traditional combinatorial algorithms do not perform well on certain problem types in this class, for instance job shop scheduling. The major difficulty is obtaining tight lower bounds which are needed to prove optimality or even optimality within a specified tolerance.

Relatively few papers and reports have been written in this area. Balas [1985] pioneered the study of scheduling polyhedra with his work on the facial structure of the job

shop scheduling problem. Queyranne [1986] completely characterized the polyhedron associated with the nonpreemptive single machine scheduling problem. Dyer and Wolsey [1990] examined several formulations for the single machine scheduling problem with release times. Queyranne and Wang [1988] generalized Queyranne's results to the nonpreemptive single machine scheduling problem with precedence constraints. Sousa and Wolsey [1989] investigated time indexed formulations for several variants of the nonpreemptive single machine scheduling problem. Finally, Wolsey [1989] compared different formulations for the single machine scheduling problem with precedence constraints.

In this paper, we propose a formulation that involves start time and sequence determining variables for the nonpreemptive single machine scheduling problem with release times and we develop a cutting plane algorithm based on this formulation and several classes of valid inequalities. The paper is organized as follows. In the next section, we formally introduce the single machine scheduling problem with release times and propose a mixed integer programming formulation. In the subsequent sections, we discuss a linear relaxation, various classes of valid inequalities, separation heuristics, and the cutting plane algorithm we have implemented. In the final sections, we present computational results and possible enhancements that are based on using additional variables and column generation.

2 The single machine scheduling problem with release times

A set J of n jobs has to be processed without interruption on a single machine that can handle at most one job at a time. Each job $j \in J$ becomes available at its *release time* r_j and requires a *processing time* p_j. The problem is to find a feasible schedule that minimizes the weighted sum of the completion times. In the sequel, we assume that both r_j and p_j are nonnegative integers and the jobs are numbered in order of nondecreasing release time, i.e., $0 \leq r_1 \leq r_2 \leq \ldots \leq r_n$.

For any ordering π of the jobs, there exists one feasible schedule that dominates all others. In this schedule, called an *active schedule*, each job is processed as early as possible, given the processing order. If $t_{\pi(j)}$ denotes the start time of job $\pi(j)$, the active schedule for π is

$$t_{\pi(1)} = r_{\pi(1)};$$

$$t_{\pi(j)} = \max(r_{\pi(j)}, t_{\pi(j-1)} + p_{\pi(j-1)}) \quad \text{for } j = 2, \ldots, n.$$

The above observation shows that we can restate the nonpreemptive single machine problem with release times as: find a permutation and associated active schedule for

which the objective function is minimum. Therefore, to obtain a valid formulation it suffices to find a linear inequality description of the set of permutations and of the active schedule associated with a given permutation.

Let δ_{ij} be equal to 1 if job i precedes j and 0 otherwise. Then $\delta \in B^{n(n-1)}$ is a permutation if and only if it satisfies

$$\begin{aligned} &\delta_{ij} + \delta_{ji} = 1 \quad 1 \le i < j \le n; \\ &\delta_{ij} + \delta_{jk} + \delta_{ki} \le 2 \quad 1 \le i \ne j \ne k \le n. \end{aligned} \tag{2.1}$$

The inequalities in (2.1) are called *triangle inequalities.* Grötschel, Jünger, and Reinelt [1984, 1985] study the convex hull of solutions to these inequalities and present a cutting plane branch and bound algorithm for finding a minimum weight permutation.

A linear description of the active schedule associated with a given permutation is provided by the following theorem. Note that $\delta_{ik} + \delta_{kj} - 1 \le \delta_{ik}\delta_{kj}$, which equals one if and only if $\delta_{ik} = \delta_{kj} = 1$.

Theorem 1 *The following linear program determines the active schedule associated with a permutation given in terms of δ-variables. (For convenience let $\delta_{jj} = 1$.)*

$$\min \sum_{1 \le j \le n} t_j$$

subject to (2.2)

$$t_j \ge r_i \delta_{ij} + \sum_{k<i, k \ne j} p_k(\delta_{ik} + \delta_{kj} - 1) + \sum_{k \ge i, k \ne j} p_k \delta_{kj} \quad 1 \le i, j \le n.$$

Proof. Clearly the active schedule associated with δ is a solution. To show that the active schedule associated with δ is an optimal solution, consider the dual linear program given by

$$\max \sum_{1 \le i,j \le n} u_{ij}(\delta_{ij} r_i + \sum_{k<i, k \ne j} p_k(\delta_{ik} + \delta_{kj} - 1) + \sum_{k \ge i, k \ne j} p_k \delta_{kj})$$

subject to

$$\sum_{1 \le i \le n} u_{ij} = 1 \qquad 1 \le j \le n;$$

$$u \in R_+^{n^2}.$$

Observe that any feasible schedule consists of a number of blocks, each consisting of a set of jobs that are processed continuously. The value of the dual solution given by

$$u_{ij} = \begin{cases} 1 & \text{if job } i \text{ is the first job in the block that contains job } j \\ 0 & \text{otherwise} \end{cases}$$

is equal to the sum of the start times of the active schedule. □

Combining (2.1) and (2.2), we obtain the following mixed integer programming formulation for the single machine scheduling problem with release times.

$$\min \sum_{1 \le j \le n} w_j t_j$$

subject to

$$\begin{aligned} & t_j \ge r_i \delta_{ij} + \sum_{k<i, k \ne j} p_k(\delta_{ik} + \delta_{kj} - 1) + \sum_{k \ge i, k \ne j} p_k \delta_{kj} \quad 1 \le i, j \le n; \\ & \delta_{ij} + \delta_{ji} = 1 \quad 1 \le i < j \le n; \\ & \delta_{ij} + \delta_{jk} + \delta_{ki} \le 2 \quad 1 \le i \ne j \ne k \le n; \\ & \delta \in B^{n(n-1)}. \end{aligned} \tag{2.3}$$

Note that it is better to put a job k with $k < i$ and $r_k = r_i$ in the second sum of the constraints defining the start time t_j of job j. However, for simplicity of presentation, we will not do this explicitly. Note also that in the absence of degeneracy, exactly one of the n constraints defining the start time t_j of job j is satisfied with equality in an optimal solution to (2.3).

3 A linear programming relaxation

The first step in the development of a cutting plane algorithm is the definition of an initial linear programming relaxation. After relaxing the integrality condition on the δ-variables in (2.3) to obtain a linear program, we made three additional modifications, one to reduce its size, the others to strengthen it.

To keep the size of the linear programming relaxation reasonable, the triangle inequalities are dropped. As a consequence, an integral solution to the linear programming relaxation may not be feasible.

An examination of the time constraints reveals that they all have the following structure. A first part that establishes a 'base' release time for the constraint, a second part that deals with all the jobs that have a release time that falls before the base release

time, and a third part that deals with all the jobs that have a release time that falls after the base release time.

Observe that if job k has a release time before the base release time, but has an earliest possible completion time after the base release time, i.e., $r_k < r_i$ and $r_k + p_k > r_i$, then $p_k(\delta_{ik} + \delta_{kj} - 1)$ is dominated by $(r_i - r_k)(\delta_{ik} + \delta_{kj} - 1) + (r_k + p_k - r_i)\delta_{kj}$. In fact, we no longer distinguish between jobs that have a release time before the base release time and jobs that have a release time after the base release time. Instead we distinguish between processing time that may fall before the base release time and processing time that must fall after the base release time. Furthermore, the base release time part $r_i\delta_{ij}$ can be strengthened by observing that for $i < j$ it is dominated by r_i and that for $i > j$ it is dominated by $r_j + (r_i - r_j)\delta_{ij}$.

Consequently, we take the initial linear programming formulation to be the following.

$$\min \sum_{1 \le j \le n} w_j t_j$$

subject to

$$\begin{aligned}
t_j \ge\ & r_i + \sum_{k<i,k\neq j,r_k+p_k\le r_i} p_k(\delta_{ik} + \delta_{kj} - 1) \\
& + \sum_{k<i,k\neq j,r_k+p_k>r_i} [(r_i - r_k)(\delta_{ik} + \delta_{kj} - 1) + (r_k + p_k - r_i)\delta_{kj}] \\
& + \sum_{k\ge i,k\neq j} p_k\delta_{kj} \qquad 1 \le i \le j \le n; \\
t_j \ge\ & r_j + (r_i - r_j)\delta_{ij} + \sum_{k<i,k\neq j,r_k+p_k\le r_i} p_k(\delta_{ik} + \delta_{kj} - 1) \\
& + \sum_{k<i,k\neq j,r_k+p_k>r_i} [(r_i - r_k)(\delta_{ik} + \delta_{kj} - 1) + (r_k + p_k - r_i)\delta_{kj}] \\
& + \sum_{k\ge i,k\neq j} p_k\delta_{kj} \qquad 1 \le j < i \le n; \\
\delta_{ij} + \delta_{ji} &= 1 \quad 1 \le i < j \le n; \\
\delta &\in R_+^{n(n-1)}.
\end{aligned} \tag{3.1}$$

4 Equal release times

When all release times are equal (without loss of generality $r_k = 0$ for $k \in J$), (3.1) reduces to

$$\min \sum_{1 \le j \le n} w_j t_j$$

subject to

$$t_j \ge \sum_{1 \le k \ne j \le n} p_k \delta_{kj} \quad 1 \le j \le n; \tag{4.1}$$

$$\delta_{ij} + \delta_{ji} = 1 \quad 1 \le i < j \le n;$$

$$\delta \in R_+^{n(n-1)},$$

and we have the following result.

Theorem 2 *When $r_k = 0$ for all $k \in J$, the optimal objective value of (4.1) equals the value of a minimum weight schedule.*

Proof. Smith's rule [Smith 1956] says that if the jobs are in order of decreasing ratio $\frac{w_j}{p_j}$, then $(1, 2, ..., n)$ is an optimal sequence giving an objective value $\sum_{1 \le j \le n} w_j (\sum_{1 \le i < j} p_i)$. The dual of (4.1) is

$$\max \sum_{1 \le i < j \le n} v_{ij}$$

subject to

$$u_j = w_j \quad 1 \le j \le n;$$

$$p_i u_j - v_{ij} \ge 0 \quad 1 \le i < j \le n;$$

$$p_j u_i - v_{ij} \ge 0 \quad 1 \le i < j \le n;$$

$$u \in R_+^n, v \in R^{\frac{n(n-1)}{2}}.$$

Taking $v_{ij} = p_i w_j$ produces a dual feasible solution, since $i < j \Rightarrow \frac{w_i}{p_i} \ge \frac{w_j}{p_j} \Rightarrow p_j w_i - p_i w_j \ge 0 \Rightarrow p_j u_i - v_{ij} \ge 0$, with objective value $\sum_{1 \le j \le n} w_j (\sum_{1 \le i < j} p_i)$. By weak duality, the optimal value of (4.1) is at least as large. □

Since (4.1) does not include the triangle inequalities, it contains infeasible integer solutions. If we add the triangle inequalities, then all solutions with integral δ are feasible. The proof of Theorem 2 shows that there is an integral optimal solution for any $w \ge 0$, and the problem is unbounded if any component of w is negative. Hence we have the following corollary.

Corollary 1 *When $r_k = 0$ for all $k \in J$, the constraints of (4.1) and the triangle inequalities give the convex hull of feasible solutions.*

Queyranne [1986] shows that in t-space the convex hull of the set of feasible schedules is defined by following system of linear inequalities

$$\sum_{j \in S} p_j t_j \geq \sum_{j \in S} p_j \sum_{i \in S, i<j} p_i \quad \forall S \subseteq J, S \neq \emptyset; \tag{4.2}$$

Proposition 1 *The linear inequalities (4.2) are nonnegative linear combinations of the inequalities (4.1).*

Proof.

$$t_j \geq \sum_{k \in J \setminus \{j\}} p_k \delta_{kj} \geq \sum_{k \in S \setminus \{j\}} p_k \delta_{kj}.$$

Hence

$$\begin{aligned} \sum_{j \in S} p_j t_j &\geq \sum_{j \in S} p_j \sum_{k \in S \setminus \{j\}} p_k \delta_{kj} \\ &= \sum_{j \in S} p_j \sum_{k \in S, k<j} p_k (\delta_{kj} + \delta_{jk}) \\ &= \sum_{j \in S} p_j \sum_{k \in S, k<j} p_k. \ \square \end{aligned}$$

5 Valid inequalities

In this section, we derive several classes of valid inequalities. The first class of valid inequalities establishes lower bounds on the position of jobs in an optimal schedule based on a well-known dominance criterion that says that if $r_i + p_i \leq r_j$, then j will not be the first job in any optimal schedule.

Dominance inequalities. *Let $\alpha_j = \min_{k \in J: r_k + p_k \leq r_j} (r_k + p_k)$, if it is defined. Then*

$$\sum_{i \in J: r_i \leq \alpha_j} \delta_{ij} \geq 1$$

is a valid inequality.

If, as we have assumed throughout, the jobs are given in order of increasing release times, it is very simple to generate all possible dominance inequalities in advance. Since the number of dominance inequalities is probably small, we can add all of them to the original formulation.

The second class of valid inequalities is derived directly from the mixed integer programming formulation. Observe that $\delta_{ik} + \delta_{kj} - 1$ can be negative based on the values of the sequence determining variables. Therefore, we consider time constraints that only use a subset of the terms that involve two sequence determining variables.

Subset inequalities. *Let r_{ij} denote the base release time, i.e., r_i if $i \leq j$ and $r_j + (r_i - r_j)\delta_{ij}$ if $i > j$, and let $S \subseteq \{1, ..., i-1\} \setminus \{j\}$, then*

$$t_j \geq r_{ij} + \sum_{k \in S, r_k + p_k \leq r_i} p_k(\delta_{ik} + \delta_{kj} - 1)$$

$$+ \sum_{k \in S, r_k + p_k > r_i} [(r_i - r_k)(\delta_{ik} + \delta_{kj} - 1) + (r_k + p_k - r_i)\delta_{kj}]$$

$$+ \sum_{i \leq k \leq n, k \neq j} p_k \delta_{kj}$$

is a valid inequality for all $i, j \in J$.

The next four classes of valid inequalities establish different base release times. After presenting them, we will indicate how they can be strengthened.

Summation inequalities I. *Let $S \subseteq J \setminus \{j\}$ and let $r_S = \min_{k \in S}\{r_k\}$ be such that $r = r_S + \sum_{k \in S} p_k > r_j$, then*

$$t_j \geq r_j + (r - r_j)(\sum_{k \in S} \delta_{kj} - |S| + 1)$$

is a valid inequality.

Proof. The validity follows from the observation that $r_S \leq r_k$ for $k \in S$ and that for any feasible sequence $\sum_{k \in S} \delta_{kj} > |S| - 1$ if and only if all jobs in S precede j. □

Sequence inequalities I. *Let $j \in S \subseteq J$ and let π be a permutation of S such that $\pi(|S|) = j$ and $r = r_{\pi(1)} + \sum_{k \in S \setminus \{j\}} p_k > r_j$, then*

$$t_j \geq r_j + (r - r_j)(\sum_{1 \leq k < |S|} \delta_{\pi(k)\pi(k+1)} - |S| + 2)$$

is a valid inequality.

Proof. The validity follows from the observation that any feasible schedule for which $\sum_{1\leq k<|S|} \delta_{\pi(k)\pi(k+1)} > |S|-2$ contains $(\pi(1),\pi(2),...,\pi(|S|))$ as a subsequence. □

Example 1
Consider the problem instance given in Table 1. The optimal sequence is (2,3,4,1) with associated start times $t_1 = 10, t_2 = 2, t_3 = 6$ and $t_4 = 8$. The solution to the initial linear programming relaxation is given in Table 2. The solution violates the summation

	r_j	p_j	w_j
1	1	4	1
2	2	3	2
3	6	2	3
4	7	2	1

Table 1: Problem instance

t_j	1	2	3	4
	8.00	2.00	6.00	8.00

δ_{ij}	1	2	3	4
1		0.00	0.25	0.25
2	1.00		1.00	1.00
3	0.75	0.00		1.00
4	0.75	0.00	0.00	

Table 2: Solution to the initial linear programming relaxation

I inequality (with $j = 3$ and $S = \{1,2\}$)

$$t_3 \geq r_3 + (r_1 + p_1 + p_2 - r_3)(\delta_{13} + \delta_{23} - 1) = 4 + 2(\delta_{13} + \delta_{23})$$

and the sequence I inequality (with $j = 3, S = \{1,2\}$ and $\pi = (2,1)$)

$$t_3 \geq r_3 + (r_2 + p_2 + p_1 - r_3)(\delta_{21} + \delta_{13} - 1) = 3 + 3(\delta_{21} + \delta_{13}).$$

The two inequalities above replace r_i in the base release time $r_j + (r_i - r_j)$ by r to get $r_j + (r - r_j)$. The two inequalities below replace r_j in the base release time to get $r + (r_i - r)$.

Summation inequalities II. *Let $S \subseteq J\backslash\{j\}$ and let $r_S = \min_{k\in S}\{r_k\}$ be such that $r_j < r = r_S + \sum_{k\in S} p_k \leq r_i$, then*

$$t_j \geq r(\sum_{k\in S} \delta_{kj} - |S| + 1) + (r_i - r)(\sum_{k\in S} \delta_{kj} + \delta_{ij} - |S|)$$

is a valid inequality.

Proof. Same as for Summation inequalities I and $t_j \geq r_i$ if $\delta_{ij} = 1$. □

Sequence inequalities II. *Let $j \in S \subseteq J$ and let π be a permutation of S such that $\pi(|S|) = j$ and $r_j < r = r_{\pi(1)} + \sum_{k \in S \setminus \{j\}} p_k \leq r_i$, then*

$$t_j \geq r\Big(\sum_{1 \leq k < |S|} \delta_{\pi(k)\pi(k+1)} - |S| + 2\Big) + (r_i - r)\Big(\sum_{1 \leq k < |S|} \delta_{\pi(k)\pi(k+1)} + \delta_{ij} - |S| + 1\Big)$$

is a valid inequality.

Proof. Same as for sequence inequalities I and $t_j \geq r_i$ if $\delta_{ij} = 1$. □

As mentioned before, the four families of valid inequalities establish different base release times. Therefore, the inequalities can be strengthened by considering the jobs $k \in J \setminus S$ that have not been used in the definition of the base release time and adding the term $(r_k + p_k - r)\delta_{kj}$ if $r_k < r$ and $r_k + p_k > r$ or the term $p_k\delta_{kj}$ if $r_k \geq r$.

Example 2
Again, consider the problem instance given in Table 1 and the solution to the initial linear programming relaxation given in Table 2. The summation II inequality (with $S = \{2\}, j = 1$ and $i = 3$) and sequence II inequality (with $S = \{1,2\}, \pi = (2,1), j = 1$, and $i = 3$) are the same, namely

$$t_1 \geq (r_2 + p_2)\delta_{21} + (r_3 - r_2 - p_2)(\delta_{21} + \delta_{31} - 1).$$

This inequality is not violated, but it can be strengthened by adding $p_3\delta_{31} + p_4\delta_{41}$ to the right hand side. The strengthened inequality is violated.

Note that all the valid inequalities we have derived so far relate start times to release times and processing times, i.e., the data that specify the problem instance. However, if at some point it is established that some job i precedes another job j, for instance if in the context of a branch and bound algorithm branching is done by variable fixing, we have the following result.

Precedence inequalities I. *If job i precedes job j, then for any $S \subseteq J \setminus \{i,j\}$*

$$t_j \geq t_i + p_i + \sum_{k \in S, k < i} p_k(\delta_{ik} + \delta_{kj} - 1) + \sum_{k \in S, k > i} p_k\delta_{kj}$$

is a valid inequality.

A natural nonlinear inequality that relates start times is $t_j \geq (t_i + p_i)\delta_{ij}$. This inequality can be linearized by replacing t_i by any lower bound l_i to obtain $t_j \geq (l_i+p_i)\delta_{ij}$. The trivial lower bound $l_i = r_i$ is useless, since it results in an inequality that is dominated by the inequalities in the original formulation. All other known lower bounds on t_i involve sequence determining variables, which again results in a nonlinear inequality. However, in this case the nonlinear terms involve precisely two sequence determining variables, and, as in the formulation (2.2), can be linearized using $\delta_{ij}\delta_{kl} \geq \delta_{ij} + \delta_{kl} - 1$.

Precedence inequalities II. *If job i precedes job j and $t_i \geq f(\delta)$, then a valid inequality is*

$$t_j \geq \overline{f}(\delta) + p_i\delta_{ij},$$

where $\overline{f}(\delta)$ is obtained from $f(\delta)\delta_{ij}$ by replacing all nonlinear terms $\delta_{ij}\delta_{kl}$ by $\delta_{ij}+\delta_{kl}-1$.

Example 3
Again, consider the problem instance given in Table 1. The solution to the initial linear programming relaxation plus the violated summation I and violated sequence I inequalities of Example 1 is given in Table 3. The solution violates the precedence II inequality

t_j	1	2	3	4
	7.86	2.21	6.42	8.00

δ_{ij}	1	2	3	4
1		0.07	0.21	0.21
2	0.93		1.00	1.00
3	0.79	0.00		1.00
4	0.79	0.00	0.00	

Table 3: Solution to the extended linear programming relaxation

$$t_4 \geq r_3\delta_{34} + (r_2 + p_2 + p_1 - r_3)(\delta_{21} + \delta_{13} + \delta_{34} - 2) + p_3\delta_{34}. \quad [t_4 \geq (t_3 + p_3)\delta_{34}]$$

where the lower bound on t_3 is given by the sequence I inequality of Example 1.

6 Separation

Any linear programming based algorithm that has to deal with an exponential number of inequalities will start with a partial description of the set of feasible solutions and will subsequently try to identify and add violated inequalities. The problem of identifying violated inequalities is known as the separation problem. Formally, if we are given a polyhedron $P \in R^n$ and a point $c \in R^n$, the *separation problem* [Grötschel, Lovasz, and

Schrijver 1981] is the one of deciding whether $c \in P$ and, if not, to find a separating hyperplane, i.e., an inequality that is satisfied by all points $c' \in P$ but violated by c.

In the remainder of this section, we will discuss the separation procedures that are implemented in our cutting plane algorithm. The solution to the current linear program is denoted by (t^*, δ^*).

The triangle inequalities. The triangle inequalities of the linear ordering polytope are handled by enumerating all $n(n-1)(n-2)/3$ of them and identifying those that are violated.

Incorporation of the following observations increase the efficiency of the enumeration. First, any permutation of three elements has a representation in which the elements are in increasing order or in decreasing order. Secondly, as soon as we detect that $\delta^*_{ij} = 0$ or $\delta^*_{ij} + \delta^*_{jk} \leq 1$, we know the inequality will not be violated.

The subset inequalities. For each of the n^2 time constraints in the original formulation, we check whether it contains terms involving two sequence determining variables that currently have a negative contribution, i.e., $\delta^*_{ik} + \delta^*_{kj} < 1$, and, if so, whether the deletion of these terms would lead to a violated inequality.

The summation inequalities I. For each job j, we try to find a violated inequality. The separation heuristic is based on two properties of a set S^* that, for a given job j, induces a summation inequality, if one exists, for which the violation is maximum.

1. If $i_1 = argmin_{k \in S^*}\{r_k\}$ and $i_2 = argmin_{k \in S^* \setminus \{i_1\}}\{r_k\}$, then $r_{i_1} + p_{i_1} > r_{i_2}$.
2. If $\delta^*_{kj} = 1$, then job k will be in S^*, unless it causes a conflict with property (1).

In order to not have to worry about property (1), we construct sets S_k for each job k that contain, besides job k itself, only jobs with a release time larger than r_k, and in the end take S to be the best among the S_k's we have constructed. The set S_k initially contains job k and all jobs l $(l > k)$ for which $\delta^*_{lj} = 1$, giving a base release time $r_j + (r - r_j)\vartheta$, with $r = r_k + \sum_{l \in S_k} p_l$ and $\vartheta = \delta^*_{kj}$. Next, we try to expand S_k by adding jobs l $(l > k)$ with $0 < \delta^*_{lj} < 1$. Observe that any job l $(l > k)$ with $0 < \delta^*_{lj} < 1$, if added, will increase r by p_l and decrease ϑ by $1 - \delta^*_{lj}$. Note that this approach does not necessarily find an optimal S^*.

The sequence inequalities I. Since $\delta_{ij} = 1 - \delta_{ji}$, in a fractional solution, it is always possible to concentrate on a sequence determining variable δ_{ij} with $0 < \delta_{ij} \leq 0.5$ and try to identify a violated inequality that, if added to the current formulation, will force that variable to go down. The other main idea embedded in the separation heuristic for the sequence inequalities is that of trying to prove a sequence is locally optimal by disproving optimality for sequences obtained from this sequence by relocating one job.

Both ideas are illustrated by the following example. Suppose that we believe that the sequence $(\pi(1), \pi(2), ..., \pi(k), \pi(k+1), \ldots, \pi(n))$ is optimal, but $\delta^*_{\pi(k)\pi(k+1)}$ is fractional. Then, by considering the subsequences $(\pi(i), \pi(i+1), \ldots, \pi(k+1), \pi(k))$ for $i = 1, ..., k-1$, we try to identify a violated sequence inequality for the sequence $(\pi(1), \pi(2), \ldots, \pi(k+1), \pi(k), \ldots, \pi(n))$ that will force $\delta_{\pi(k+1)\pi(k)}$ to go down.

We consider three candidates for an optimal sequence: (1) the sequence associated with the best feasible schedule, (2) the sequence suggested by the current values of the start time variables, (3) the sequence suggested by the current values of the sequence determining variables, i.e., $\pi(i) < \pi(k)$ if $\sum_j \delta_{ij} < \sum_j \delta_{kj}$. (Note that the last two sequences are not necessarily the same.)

The summation inequalities II. The separation heuristic is similar to the one described for the summation inequalities I.

The sequence inequalities II. Based on the sequence associated with the best feasible schedule found so far, we enumerate all possible sets S that generate a release time r that satisfies the restrictions $r_j < r \leq r_i$.

The precedence inequalities I. If job i precedes job j, we try to identify a violated inequality by taking the sum of all terms $p_k(\delta^*_{ik} + \delta^*_{kj} - 1)$ for $k < i$ and $p_k\delta^*_{kj}$ for $k > i$ and comparing it to $t^*_j - (t^*_i + p_i)$.

The precedence inequalities II. For each of the three sequence defined above, we establish whether it contains a pair of consecutive jobs i and j, with $\pi(i) < \pi(j)$ and such that $t_i + p_i > t_j$. If so, we linearize the precedence constraint $t_j \geq (t_i + p_i)\delta_{ij}$ using one of the inequalities in the current formulation that defines t_i and that is tight with respect to the current LP solution. Then we see whether the resulting inequality is violated.

7 The algorithm

Since even for moderately sized problem instances, the number of variables and the number of constraints in the initial linear programming relaxation is rather large, we reduce both by replacing all occurrences of δ_{ji} with $j > i$ by $1 - \delta_{ij}$ and delete all equality constraints. This reduces the number of variables from n^2 to $n(n+1)/2$ and the number of constraints from $n(n-1)/2 + n^2$ to n^2.

The algorithm uses a combination of cutting planes, primal heuristics and branch and bound. In each node of the branch and bound tree the following steps are performed.

1. Solve the current linear program. If its solution is integral and satisfies the triangle inequalities, then, if necessary, modify the best primal solution found so far, fix variables based on their reduced costs and try to fathom nodes of the branch and

bound tree. If all nodes are fathomed, then stop else select another node and go to step 1.

2. Calculate the active schedule associated with the sequence suggested by the current values of the start time variables and the active schedule associated with the sequence suggested by the current values of the sequence determining variables. If necessary, modify the best primal solution found so far, fix variables based on their reduced costs and try to fathom nodes of the branch and bound tree. If all nodes are fathomed, then stop. If the current node was fathomed, then select another node and go to step 1.

3. Call the separation heuristic for the triangle inequalities to check if the current solution violates any of them. If any violated triangle inequalities are found, add them to the current linear program and go to step 1.

4. Call the separation heuristics for the subset inequalities, the summation inequalities (I and II), the sequence inequalities (I and II), and the precedence inequalities (only I) to identify if the current solution violates any of them. If any violated inequalities are found, add them to the current linear relaxation and go to step 1.

5. Branch by selecting the fractional variable δ_{ij} that is closest to one-half. On one branch $\delta_{ij} = 0$ and on the other $\delta_{ij} = 1$.

An important consequence of fixing sequence determining variables when branching, besides being able to look for violated precedence inequalities, is that we can modify a release time. If $\delta_{ij} = 1$, then $r'_j = \min(r_{j+1}, \max(r_j, r_i + p_i))$. The min operation is used to ensure that the jobs remain in order of increasing release times.

8 Computational results

The purpose of the computational study is to investigate the feasibility of using mixed-integer programming, in particular, a formulation with sequence determining as well as start time variables and an exponential number of constraints, to solve the single machine scheduling problem with release times.

The algorithm is implemented using MINTO, a tool for solving mixed integer programming problems. The heart of MINTO is a linear programming based branch and bound algorithm. Although MINTO can be used as a general purpose mixed integer optimizer, it also facilitates the development of a special purpose mixed integer optimizer since it provides mechanisms to incorporate problem specific functions. For further information on MINTO, we refer to Nemhauser, Savelsbergh, and Sigismondi [1991].

Test problems are randomly generated by a commonly used scheme. The weights and processing times are integers uniformly distributed in [1,...,10] and [1,...,5] respectively. The release times are uniformly distributed in $[0,...,\alpha\sum_{1\leq j\leq n} p_j]$, where n is the number of jobs and α a control parameter, which is usually taken between 0.3 and 0.7. We have used $\alpha = 0.5$ for all our experiments.

The actual computational study consisted of two parts. First, a general evaluation of the proposed method. Second, an evaluation of the value of the scheduling inequalities in proving optimality.

Tables 4 and 5 present the computational results for instances with 20 and 30 jobs. Several observations can be made regarding these results. The number of evaluated nodes is small and does not seem to increase very much when the number of jobs increases. The integrality gap, i.e., the difference between the value of the optimal solution and the value of the the solution to the initial linear programming relaxation, is also small (less than 4 percent for $n = 20$ and less than 3 percent for $n = 30$) and does not seem to increase with the problem size. The number of linear programs solved and the number of cuts generated is relatively large and sharply increases with problem size. Furthermore, the linear programs become harder and harder to solve when the number of generated inequalities increases. The first two observations are positive, whereas the last two observations are negative.

problem	z_{OPT}	z_{LP}	gap	#nodes	#LPs	#triangle cuts	#scheduling cuts
1	2839	2738.95	3.52	2	16	141	85
2	3915	3891.26	0.60	3	22	93	66
3	4750	4708.16	0.88	4	38	190	68
4	4428	4364.09	1.44	2	15	120	41
5	3113	3029.90	2.66	17	72	164	147
6	3437	3412.10	0.72	32	108	185	282
7	3305	3254.42	1.53	2	11	96	24
8	3287	3213.00	2.25	27	161	364	519
9	3172	3130.81	1.29	1	5	75	21
10	3530	3498.20	0.90	56	247	343	853

Table 4: Computationals results for $n = 20$.

To evaluate the value of the scheduling inequalities in proving optimality, we have solved the instances with 20 jobs with a bare-bone version of the algorithm. This bare-bone version of the algorithm generates only triangle inequalities. The results shown in Table 6 clearly demonstrate the value of the scheduling inequalities. In all cases the number of evaluated nodes, and the number of linear programs that have to be solved,

problem	z_{OPT}	z_{LP}	gap	#nodes	#LPs	#triangle cuts	#scheduling cuts
1	8352	8251.86	1.19	36	155	557	382
2	6653	6567.91	1.27	18	98	694	341
3	8359	8277.05	0.98	12	66	341	572
4	7116	7071.96	0.61	3	21	268	38
5	8859	8672.28	2.10	8	63	716	172
6	8408	8264.84	1.70	19	102	987	270
7	8156	8004.00	1.86	30	180	604	693
8	7653	7583.83	0.90	11	62	324	144
9	7235	7149.73	1.17	52	286	966	1972
10	7096	7005.68	1.27	94	426	848	1424

Table 5: Computationals results for $n = 30$.

increases sharply when no scheduling inequalities are generated.

problem	z_{OPT}	z_{LP}	#nodes	#LPs solved
1	2839	2738.95	54	130
2	3915	3891.26	15	47
3	4750	4708.16	56	126
4	4428	4364.09	15	40
5	3113	3029.90	26	54
6	3437	3412.10	75	162
7	3305	3254.42	28	71
8	3287	3213.00	208	426
9	3172	3130.81	4	10
10	3530	3498.20	128	274

Table 6: Computationals results for the bare-bone version for $n = 20$.

The results show that the mixed-integer programming formulation is strong but many cuts are needed to prove optimality. Thus the approach is successful on relatively small problems but further work is required to make it competitive with purely combinatorial methods or to achieve the impressive computational results that have been obtained with cutting plane branch and bound algorithms for problems such as the traveling salesman problem [Padberg and Rinaldi 1987, Padberg and Rinaldi 1988, Grötschel and Holland 1988].

The efficiency of our cutting plane branch and bound approach can be improved in at least three ways.

First, we can improve the current implementation. There are several possibilities here. The most promising ones relate to solving the LP relaxation more efficiently. It currently consumes 85 percent of the computation time because it is unnecessarily large. A better approach to solving the LP relaxation would be to fix most sequence determining variables and ignore most constraints temporarily using the best feasible solution as a guide. In particular, only the sequence determining variables and constraints that are locally relevant with respect to the current best solution would be active, the remaining sequence determining variables would be fixed at their values in the current best solution and the remaining constraints would be ignored. The temporarily fixed variables would be activated if a better feasible solution or if their reduced costs indicated doing so. The ignored constraints would be judiciously checked by an implicit enumeration separation routine.

Second, we can make use of the objective function. Again, there are various possibilities. We can use bounds and feasible solutions that arise from combinatorial methods [Harari and Potts 1983], or we can use additional dominance relations [Rinaldi and Sassano 1977].

Finally, we can improve the mixed-integer formulation. Two possible ways of accomplishing this are given in the final section.

9 Extended formulations using additional variables

Improved linearization

In our basic model, see (2.2) or (3.1), we linearized the term $\delta_{ik}\delta_{kj}$ by the lower bound $\delta_{ik} + \delta_{kj} - 1$ which may be negative. A better approximation is obtained by introducing the 0-1 variables δ_{ikj} for all $i \neq j \neq k$ and replacing the terms $p_k(\delta_{ik} + \delta_{kj} - 1)$ by $p_k\delta_{ikj}$ where in the linear programming relaxation we add the constraints

$$\delta_{ikj} \geq \delta_{ik} + \delta_{kj} - 1,$$

$$\delta_{ikj} \geq 0,$$

$$\delta_{ikj} \leq \delta_{ik},$$

$$\delta_{ikj} \leq \delta_{kj}.$$

Now observe that because of the cost structure there is an optimal solution with $\delta_{ikj} = \max(0, \delta_{ik} + \delta_{kj} - 1)$. Thus the upper bound constraints are superfluous.

The idea is to use the δ_{ikj} in place of the subset inequalities. In particular, suppose $\delta_{ik} + \delta_{kj} - 1 < 0$ and there is a violated subset inequality that contains this negative term. Instead of adding it, we strengthen all of the original inequalities containing $\delta_{ik} + \delta_{kj} - 1$

by replacing $\delta_{ik} + \delta_{kj} - 1$ by δ_{ikj}. The weaker versions can be removed. Now with $\delta_{ikj} = 0$, the strengthened version of the inequality will be violated.

The advantage of this approach is that we can accomplish with at most $O(n^3)$ additional variables what could require an exponential number of subset constraints. Moreover, the additional variables can be generated as we need them. The disadvantage is that a much more complicated implementation is required.

Block variables

We have already noted that, in the absence of degeneracy, exactly one of the constraints of the original formulation defining the start time t_j of job j is satisfied with equality in an optimal solution to (2.3), namely the one associated with the first job of the block that contains job j. These first jobs of the blocks, called *block-headers*, have another, maybe even more important, property: their start times are exactly equal to their release times. Thus, if we could identify a block-header, we can fix its start time by providing an upper bound as well as a lower bound. More generally, as seen in the formulation below, the block variables can be used to get upper bound constraints on start times which may be violated by fractional solutions in the formulation (2.3).

To identify block-headers we introduce variables u_{ij} equal to 1 if job i is the header of the block that contains job j and 0 otherwise and we let T_j be an upper bound on t_j over all solutions that are candidates for optimality. This leads to the following formulation

$$\min \sum_{1 \le j \le n} w_j t_j$$

subject to

$$t_j \ge r_i u_{ij} + \sum_{k<i, k \ne j} p_k (u_{ik} + \delta_{kj} - 1) + \sum_{k \ge i, k \ne j} p_k \delta_{kj} \qquad 1 \le i, j \le n;$$

$$t_j \le r_i + \sum_{k<i, k \ne j} p_k (u_{ik} + \delta_{kj})/2 + \sum_{k \ge i, k \ne j} p_k \delta_{kj} + (1 - u_{ij})(T_j - r_i) \qquad 1 \le i \ne j \le n;$$

$$t_j \le r_j + (1 - u_{jj})(T_j - r_j) \qquad 1 \le j \le n;$$

$$\delta_{ij} + \delta_{ji} = 1 \qquad 1 \le i < j \le n;$$

$$\delta_{ij} + \delta_{jk} + \delta_{ki} \le 2 \qquad 1 \le i \ne j \ne k \le n;$$

$$\sum_{1 \le i \le n} u_{ij} = 1 \qquad 1 \le j \le n;$$

$$u_{ij} \le u_{ii} \qquad 1 \le i, j \le n;$$

$$u_{ij} \leq \delta_{ij} \qquad 1 \leq i \neq j \leq n;$$

$$\delta \in B^{n(n-1)};\ u \in B^{n^2}.$$

Observe that if we replace u_{ij} by δ_{ij} in the inequalities that provide lower bounds on the start times and delete all other inequalities involving block variables, we obtain formulation (2.3).

The advantage of the block oriented formulation over the original formulation is the presence of upper bounds. However, the linear relaxation of the upper bound inequalities for the block-headers is relatively weak. Several classes of valid inequalities have been derived to force the block variables to their proper values. One class follows from the observation that the constraints $\sum_{1 \leq i \leq n} u_{ij} = 1$, $u_{ij} \leq u_{ii}$, and $u \in B^{n^2}$ are precisely those that appear in the well-known uncapacitated facility location problem (UFL), so that polyhedral results for UFL can be immediately applied (see, e.g., Cornuejols, Nemhauser, and Wolsey [1990]). In addition summation inequalities and sequence inequalities similar to those derived for the δ-variables also can be derived here. The efficacy of these ideas as well as those mentioned at the end of the last section is being explored and will be reported in a sequel to this paper.

10 References

E. BALAS (1985). On the facial structure of scheduling polyhedra. *Mathematical Programming Study 24*, 179-218

G. CORNUEJOLS, G.L. NEMHAUSER, L.A. WOLSEY (1990). The uncapacitated facility location problem. P.B. MIRCHANDANI, R.L. FRANCIS (eds.). *Discrete Location Theory*, Wiley, Chichester.

H. CROWDER, E.L. JOHNSON, M.W. PADBERG (1983). Solving large-scale zero-one linear programming problems. *Oper. Res. 31*, 803-834.

M.E. DYER, L.A. WOLSEY (1990). Formulating the single machine sequencing problem with release dates as a mixed integer program. *Discrete Applied Mathematics 26*, 255-270.

M. GRÖTSCHEL, O. HOLLAND (1988). *Solution of Large-Scale Symmetric Travelling Salesman Problems.* Report No.73, Universität Augsburg.

M. GRÖTSCHEL, M. JÜNGER, G. REINELT (1984). A cutting plane algorithm for the linear ordering problem. *Oper. Res. 32*, 1195-1220.

M. GRÖTSCHEL, M. JÜNGER, G. REINELT (1985). Facets of the Linear Ordering Polytope. *Math. Programming 33*, 43-61.

M. GRÖTSCHEL, L. LOVASZ, A. SCHRIJVER (1981). The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica 1*, 161-197.

M. GRÖTSCHEL, M.W. PADBERG (1985a). Polyhedral theory. E.L. LAWLER, J.K. LENSTRA, A.H.G. RINNOOY KAN, D.B. SHMOYS (eds.). *The traveling salesman problem: a guided tour of combinatorial optimization*, Wiley, Chichester.

M. GRÖTSCHEL, M.W. PADBERG (1985b). Polyhedral computations. E.L. LAWLER, J.K. LENSTRA, A.H.G. RINNOOY KAN, D.B. SHMOYS (eds.). *The traveling salesman problem: a guided tour of combinatorial optimization*, Wiley, Chichester.

A.M.A. HARARI, C.N. POTTS (1983). An algorithm for single machine sequencing with release time dates to minimize total weighted completion time. *Discrete Appl. Math. 5*, 99-109.

K. HOFFMAN, M. PADBERG (1985). LP-based combinatorial problem solving. *Annals of Oper. Res. 4*, 145-194.

M. JÜNGER (1985). *Polyhedral combinatorics and the acyclic subdigraph problem.* Heldermann Verlag, Berlin.

G.L. NEMHAUSER, M.W.P. SAVELSBERGH, G.C. SIGISMONDI (1991). *Functional description of MINTO, a Mixed INTeger Optimizer.*

G.L. NEMHAUSER, L.A. WOLSEY (1988). *Integer and Combinatorial Optimization.* Wiley, Chichester.

M. PADBERG, G. RINALDI (1987). Optimization of a 532-city symmetric traveling salesman problem. *Oper. Res. Letters 6*, 1-7.

M. PADBERG, G. RINALDI (1988). *A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems.* Research Report R.247, IASI-CNR, Rome.

M. QUEYRANNE (1986). *Structure of a simple scheduling polyhedron.* Working paper, University of British Columbia, Vancouver.

M. QUEYRANNE, Y. WANG (1988). *Single machine scheduling polyhedra with precedence constraints.* Working paper No. 88-MSC-017, University of British Columbia, Vancouver.

G. RINALDI, A. SASSANO (1977). *On a Job Scheduling Problem with Different Ready Times: Some Properties and a New Algorithm to Determine the Optimal Solution.* Report R.77-24, Instituto di Automatica, Universitá di Roma.

W.E. SMITH (1965). Various optimizers for single-stage production. *Naval Res. Logist. Quart. 3*, 59-66.

G. SOUSA, L.A. WOLSEY (1989). *Time Indexed Formulations of Non-Preemptive Single-Machine Scheduling Problems.* CORE discussion paper 8904, Catholic University of Louvain, Louvain-la-Neuve.

L.A. WOLSEY (1989). *Formulating Single Machine Scheduling Problems with Precedence Constraints.* CORE discussion paper 8924, Catholic University of Louvain, Louvain-la-Neuve.

EINDHOVEN UNIVERSITY OF TECHNOLOGY
Department of Mathematics and Computing Science
PROBABILITY THEORY, STATISTICS, OPERATIONS RESEARCH AND SYSTEMS THEORY
P.O. Box 513
5600 MB Eindhoven - The Netherlands
Secretariate: Dommelbuilding 0.03
Telephone: 040 - 47 3130

List of COSOR-memoranda - 1991

Number	Month	Author	Title
91-01	January	M.W.I. van Kraaij W.Z. Venema J. Wessels	The construction of a solution strategy for manpower planning problems.
91-02	January	M.W.I. van Kraaij W.Z. Venema J. Wessels	Support for problem formulation and evaluation in manpower planning problems.
91-03	January	M.W.P.Savelsbergh	The vehicle routing problem with time windows: minimizing route duration.
91-04	January	M.W.I. van Kraaij	Some considerations concerning the problem interpreter of the new manpower planning system formasy.
91-05	Febr.	G.L. Nemhauser M.W.P.Savelsbergh	A cutting plane algorithm for the single machine scheduling problem with release times.