

Decompositions, Network Flows and a Precedence Constrained Single Machine Scheduling Problem

François MARGOT

*Department of Mathematics, University of Kentucky
Lexington, KY, U.S.A.*

Maurice QUEYRANNE

*Faculty of Commerce, University of British Columbia
Vancouver, B.C., Canada*

and Yaoguang WANG

PeopleSoft, Inc., Pleasanton, CA, USA

August 14, 2000

(Revised January 2003)

Abstract

We present an in-depth theoretical, algorithmic and computational study of a linear programming (LP) relaxation to the precedence constrained single machine scheduling problem $1|prec|\sum_j w_j C_j$ to minimize a weighted sum of job completion times. On the theoretical side, we study the structure of tight parallel inequalities in the LP relaxation and show that *every* permutation schedule which is consistent with Sidney's decomposition has total cost no more than twice the optimum. On the algorithmic side, we provide a parametric extension to Sidney's decomposition and show that a finest decomposition can be obtained by essentially solving a parametric minimum cut problem. Finally, we report results obtained by an algorithm based on these developments on randomly generated instances with up to 2,000 jobs.

Subject classifications: Network/graphs, flow algorithms: parametric flows and Sidney decompositions. Production/scheduling, approximations: 2-approximation algorithm. Programming, integer, algorithms, relaxation/subgradient: integer formulation.

1 Introduction

We consider the following single machine scheduling problem, denoted $1|\text{prec}|\sum_j w_j C_j$ in the scheduling literature (see, e.g., the extensive survey by Lawler et al. (1993)): a set N of n jobs is to be processed non preemptively on a single machine, which can process only one job at a time. Associated with each job j are a positive processing time p_j and a nonnegative weight w_j . A feasible job schedule must obey a partial order specified by an acyclic graph $G = (N, \mathcal{A})$. The objective is to find a feasible sequence (or *schedule*) of jobs which minimizes the weighted sum $\sum_{j \in N} w_j C_j$ of completion times. (The basic definitions and notations can be found in Section 2).

This scheduling problem is NP-Complete (Lawler 1978) and has a long history, see Lawler et al. (1993) for references. It is a very basic problem in scheduling theory and it appears as a subproblem in more elaborate settings. An attractive idea for solving this problem is to use a decomposition technique introduced by Sidney (1975): Identify job subsets such that an optimal schedule can be obtained from pasting together optimal schedules for the job subsets. One of the goals of this paper is to give an efficient way to compute the finest such decomposition. We shall provide a detailed exposition of and extensions to Sidney's results in Section 3 below.

In this paper we also study structural, algorithmic and computational properties of a linear programming (LP) relaxation which uses as its only decision variables the job completion times C_j . This type of formulation was first introduced by Balas (1985), and studied by several authors. In particular, Queyranne and Wang (1991a) presented an extensive polyhedral study of the LP relaxation determined mainly by two families of valid inequalities, called *parallel* inequalities and *series* inequalities. Wolsey (1990) later extended the formulation with $O(n^2)$ sequence-determining binary variables, and proved that this formulation is at least as tight as

the one given by parallel and series inequalities. See the survey by Queyranne and Schulz (1996) for an exposition and references.

The linear programming formulation considered in this paper, introduced by Queyranne and Wang (1991b), was used by Schulz (1996) and Hall et al. (1997) to design a polynomial-time 2-approximation algorithm for this scheduling problem. In Section 4, we study the structure of tight parallel inequalities in this LP formulation. We show that, if the problem is not amenable to Sidney's decomposition, then a single, global parallel inequality suffices in the LP formulation. As a consequence, we also get that *every* permutation schedule which is consistent with Sidney's decomposition has total cost no more than twice the optimum, a surprising extension to Schulz's 2-approximation result.

In Section 3, we provide a parametric extension to Sidney's decomposition, and show that a finest decomposition can be obtained by essentially solving a parametric minimum cut problem, which, by Gallo, Grigoriadis and Tarjan (1989) (see also McCormick 1998), requires about the time of a single maximum flow calculation. In Section 5 we show that, after this Sidney decomposition, the LP formulation can be solved as essentially a single dual minimum cost flow problem. Both network flow problems are on networks with the jobs as nodes (plus one source and one sink), and the non-redundant precedence constraints as arcs (plus one source or sink arc per job).

In Section 6, we report results obtained by an implementation taking advantage of these theoretical developments on randomly generated instances with up to 2,000 jobs. By repeatedly applying the extended Sidney decomposition procedure and using series decompositions, most of these problems end up as a collection of subproblems, the largest of which contains approximately 60% of the jobs of the initial problem. In short computing time (determined primarily by the performance of the network flow algorithms) we obtain feasible solutions and lower bounds with an average optimality gap of about 1%, and which never exceeded 3.5%.

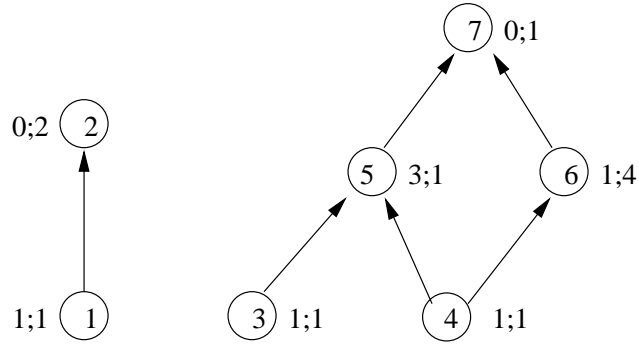


Figure 1: A precedence graph on 7 jobs; numbers $a; b$ next to node i correspond to the pair $w_i; p_i$

2 Basic notations and definitions

Throughout this paper, we use the following notation:

$w \in \mathfrak{R}_+^N$ for the row vector of the nonnegative job weights,

$p \in \mathfrak{R}_+^N$ for the row vector of the positive job processing times,

$C \in \mathfrak{R}^N$ for a column vector of job completion times,

where w and p are given and C are the decision variables. A subset J of the job set N is *proper* if $J \neq N$; it is *nontrivial* if it is proper and nonempty.

Let $G = (N, \mathcal{A})$ be an acyclic digraph that represents the precedence relations among the jobs in N . (See Figure 1). For a job subset $J \subseteq N$, let

$$\delta^+(J) = \{ij \in \mathcal{A} : i \in J, j \in N \setminus J\} \quad \text{and} \quad \delta^-(J) = \{ji \in \mathcal{A} : i \in J, j \in N \setminus J\}.$$

A subset $I \subseteq N$ is *initial* (in N) if $\delta^-(I) = \emptyset$; equivalently, if $ij \in \mathcal{A}$ and $j \in I$ imply $i \in I$. Similarly, a subset $T \subseteq K$ is *terminal* if $\delta^+(T) = \emptyset$; equivalently, if $i \in T$ and $ij \in \mathcal{A}$ imply $j \in T$. In Figure 1, the set $\{1, 3, 4, 5\}$ is initial and the set $\{6, 7\}$ is terminal. Initial subsets are also known, among others, as (order) ideals (Davey and Priestley 1990), selections (Lawler 1976), and closures (Picard 1976), whereas terminal sets are also known as filters (Davey and

Priestley 1990). Let \mathcal{I} denote the collection of all initial sets (in N), and \mathcal{T} that of all terminal sets. From these definitions, it follows immediately that

- N and \emptyset are each an initial and a terminal set;
- a subset $I \subseteq N$ is initial if and only if its complement $N \setminus I$ is terminal;
- \mathcal{I} and \mathcal{T} are a sublattices of 2^N , that is, each is closed under set union and intersection; i.e., the union $U \cup V$ and intersection $U \cap V$ of any initial (resp., terminal) sets U and V are also initial (resp., terminal) sets.

We assume throughout that $G = (N, \mathcal{A})$ does not contain a directed Hamiltonian path and that $w_j > 0$ for some $j \in N$, for otherwise the scheduling problem would be trivial.

The problem $1|prec|\sum w_j C_j$ can now be stated as a disjunctive linear program as follows:

$$\begin{aligned}
 (1) \quad z^* &= \min \quad wC \\
 \text{s.t.} \quad & C_j - C_i \geq p_j, & \forall ij \in \mathcal{A} \\
 & C_j - C_i \geq p_j \vee C_i - C_j \geq p_i, \quad \forall i \neq j, \quad ij \notin \mathcal{A}, \\
 & C \geq p.
 \end{aligned}$$

This problem is known to be strongly NP hard. In the sequel, we denote the problem (1) by $P(w, p, N, \mathcal{A})$. As $w \geq 0$, the optimal schedule is obtained as a permutation of the jobs, processed with no idle time.

For a set $J \subseteq N$, we use $w(J)$ for $\sum_{j \in J} w_j$ and $p(J)$ is defined similarly. The weight vector w is *tailing off* if there exists some terminal set $T \in \mathcal{T}$ such that $w(T) = 0$; in such a case, the scheduling problem reduces to one defined on the job set $N \setminus T$, and we may schedule the jobs in T in an arbitrary feasible sequence after all those in $N \setminus T$ without affecting the objective function value. In Figure 1, w is tailing off as the terminal set $T = \{2, 7\}$ has $w(T) = 0$.

For any nonempty job subset J , define $\rho(J) = w(J)/p(J)$. The problem $P(w, p, N, \mathcal{A})$ is *stiff* if $\rho(I) < \rho(N)$ for all proper initial subsets I of N ; otherwise, it is *Sidney decomposable*.

In Figure 1, for the initial set $I = \{3, 4, 5\}$ we have $\rho(I) = 5/3 > \rho(N) = 7/11$ and thus this instance is not stiff.

A simple result that will be used several times is the following:

Lemma 2.1 *Let $a, b \geq 0$ and $c, d > 0$. Then $\frac{a+b}{c+d} < \frac{b}{d}$ if and only if $\frac{a}{c} < \frac{b}{d}$. (A similar statement holds for $>$ and $=$ operators.)*

Proof: By cross multiplication. □

A reader interested in Sidney decompositions may now continue with Section 3 below, whereas one more immediately interested in properties of the LP relaxation of the scheduling problem may now proceed directly to Section 4.

3 Sidney Decompositions

In this section, we study extensions, properties, and algorithmic aspects of Sidney decompositions. Finding a Sidney decomposition will allow us to decompose an instance of the scheduling problem into smaller, more manageable instances.

3.1 Parametric Sidney Decomposition

We begin with a parametric extension to Sidney's decomposition theorem in Sidney (1975). In addition to its algorithmic implications, this extension also provides a short proof of Sidney's main result. For nonnegative real number λ and job set H , define $f_\lambda(H)$ by

$$f_\lambda(H) = w(H) - \lambda p(H).$$

Theorem 3.1 *Consider any fixed $\lambda \geq 0$ and any initial subset J in N such that $\rho(J) = \max\{f_\lambda(I) : I \in \mathcal{I}\}$. Then there exists an optimal schedule in which J precedes $N \setminus J$.*

Proof: As in the proof of Lemma 3 in Sidney (1975), for $\mathcal{A}' = \mathcal{A} \setminus \delta^+(J)$, we consider the related problem $P(w, p, N, \mathcal{A}')$; that is, we eliminate all precedence relations from J to $N \setminus J$. Thus, $P(w, p, N, \mathcal{A}')$ is a relaxation of $P(w, p, N, \mathcal{A})$.

Lemma 3.2 $J \in \arg \max\{f_\lambda(I) : I \text{ is initial in } (N, \mathcal{A}')\}$.

Proof: First, note that J is initial in (N, \mathcal{A}') . Now, to get a contradiction, assume that there is an initial set I in (N, \mathcal{A}') with $f_\lambda(I) > f_\lambda(J)$. Let $U = J \cup I$ and $V = J \cap I$, so U and V are initial sets in (N, \mathcal{A}') and $f_\lambda(U) + f_\lambda(V) = f_\lambda(J) + f_\lambda(I) > 2f_\lambda(J)$. But note that U and V are also initial in (N, \mathcal{A}) : this is trivial for V as it is contained in J , and \mathcal{A} and \mathcal{A}' coincide on J ; regarding U , if $jk \in \mathcal{A}$ and $k \in U$, then either $jk \in \mathcal{A}'$ implying $j \in U$, or $jk \in \mathcal{A} \setminus \mathcal{A}'$ implying $j \in J \subset U$. Since U and V are initial in (N, \mathcal{A}) , we have $f_\lambda(U) \leq f_\lambda(J)$ and $f_\lambda(V) \leq f_\lambda(J)$, a contradiction. \square

Lemma 3.3 *There exists an optimal schedule for (N, \mathcal{A}') in which J precedes $N \setminus J$.*

Proof: As in Sidney (1975), consider an optimal permutation α on (N, \mathcal{A}') and write $\alpha = (\alpha|G_1, \alpha|H_1, \alpha|G_2, \dots, \alpha|H_r, \alpha|G_{r+1})$, where $\alpha|K$ denotes the restriction of α to a subset K , such that $J = \bigcup_i H_i$, and where G_1 or G_{r+1} may be empty. By the Adjacent String Interchange Lemma (Lemma 2 in Sidney 1975), we have $\rho(H_1) \geq \rho(G_2) \geq \dots \geq \rho(H_r)$, for otherwise we could exchange a subset with its predecessor in the list, obtaining a feasible permutation for (N, \mathcal{A}') with lower total cost. Note that $\rho(H_r) \geq \lambda$, for otherwise $f_\lambda(J \setminus H_r) = f_\lambda(J) - (w(H_r) - \lambda p(H_r)) > f_\lambda(J)$ a contradiction with Lemma 3.2, since $J \setminus H_r$ is initial in (N, \mathcal{A}') . On the other hand, since H_1 is initial in (N, \mathcal{A}') , we have $\rho(H_1) \leq \lambda$. In addition, if $G_1 \neq \emptyset$, we also have $\rho(G_1) \leq \lambda$ for otherwise $f_\lambda(J \cup G_1) = f_\lambda(J) + (w(G_1) - \lambda p(G_1)) > f_\lambda(J)$ again contradicting Lemma 3.2, since $J \cup G_1$ is initial in (N, \mathcal{A}') . Therefore $\rho(G_1) = \rho(H_1) = \rho(G_2) = \dots = \rho(H_r) = \lambda$, where the first equality holds provided G_1 is non-empty. This implies that we can exchange each H_i with all preceding G_j 's, obtaining an optimal permutation for (N, \mathcal{A}') as stated in the lemma. \square

Theorem 3.1 now follows immediately, since an optimal schedule as in Lemma 3.3 is feasible for (N, \mathcal{A}) , and problem $P(w, p, N, \mathcal{A}')$ is a relaxation of $P(w, p, N, \mathcal{A})$. \square

Let $\rho^* = \max\{\rho(I) : I \in \mathcal{I}\}$. Note that Sidney's result follows as a special case, using $\lambda = \rho^*$: As $f_{\rho^*}(J) \leq 0$ for all initial sets J , and is equal to 0 if and only if $\rho(J) = \rho^*$, we have:

Corollary 3.4 *If $J \subset N$ is an initial set with $\rho(J) = \rho^*$ then there exists an optimal schedule for (N, \mathcal{A}) in which J precedes $N \setminus J$.*

3.2 An Algorithm for Parametric Sidney Decomposition

We now discuss an algorithmic implication of Theorem 3.1. As observed by Picard (1976) and Lawler (1976), Lawler (1978), finding an initial set J which maximizes $f_\lambda(J)$ is equivalent to finding a minimum s, t -cut in the network $\mathcal{N}_\lambda = (N^*, \mathcal{A}^*, c_\lambda)$ where $N^* = N \cup \{s, t\}$ and $s, t \notin N$ are a *source* and a *sink*; $\mathcal{A}^* = \mathcal{A} \cup \{sj, jt : j \in N\}$ and $c_\lambda : \mathcal{A}^* \mapsto \mathfrak{R}_+$ is defined by $c_\lambda(ij) = +\infty$ for all $ij \in \mathcal{A}$,

$$c_\lambda(si) = \max\{-w_i + \lambda p_i, 0\} \quad \text{and} \quad c_\lambda(it) = \max\{w_i - \lambda p_i, 0\}$$

for all $i \in N$. Indeed, by construction of the graph (N^*, \mathcal{A}^*) , every subset $J \subset N^*$ with $s \notin J$, $t \in J$ and finite cut capacity $c_\lambda(\delta_{N^*}^-(J)) < +\infty$ defines an initial set $I = J \setminus \{t\}$ in N with $f_\lambda(I) = c_\lambda(\delta_{N^*}^-(\{t\})) - c_\lambda(\delta_{N^*}^-(J))$. That is, the sink side of the minimum cut identified by the flow defines an initial set in N maximizing f_λ . Note that (because $w \geq 0$) N is optimal for $\lambda = 0$ and (since $p > 0$) \emptyset is optimal for all $\lambda \geq \max\{w_j/p_j : j \in N\}$.

As the parameter $\lambda \geq 0$ increases, we obtain a parametric minimum cut problem where arcs adjacent to the source have monotonically non-decreasing capacity, those adjacent to the sink have monotonically non-increasing capacity and other arcs have constant capacity. This is precisely the setting for the parametric maximum flow algorithm, hereafter called the GGT algorithm, due to Gallo, Grigoriadis and Tarjan (1989). Indeed, the GGT algorithm produces, in about the time needed to compute a single maximum flow on a network \mathcal{N}_λ , a nested family of subsets $\emptyset = H_0 \subset H_1 \subset \dots \subset H_k = N$ and a sequence of *breakpoints* $+\infty = \lambda_0 > \lambda_1 > \dots > \lambda_k \geq \lambda_{k+1} = 0$ such that for all $i = 0, \dots, k$, H_i maximizes f_λ for all values of λ in the interval $[\lambda_{i+1}, \lambda_i]$. (Note that we may have $\lambda_k = \lambda_{k+1} = 0$ if w is tailing off, as defined at the end of Section 2.)

Proposition 3.5 *Let $\emptyset = H_0 \subset H_1 \subset \dots \subset H_k = N$ be a nested sequence of initial sets constructed by the GGT algorithm. For $i = 1, \dots, k$ define $J_i = H_i \setminus H_{i-1}$. Then there exists an optimal schedule for $P(w, p, N, \mathcal{A})$ in which J_i precedes J_{i+1} for all $i = 1, \dots, k - 1$.*

This Proposition shows that we can use the GGT algorithm to identify a decomposition of the job set N such that an optimal schedule is obtained from optimal schedules for the job subsets in the decomposition. In the rest of this subsection, we prove this Proposition and, in the next subsection, we show how it relates to Sidney's original decomposition. In particular, we shall need to consider special cases where there are several optimal initial sets. The reader not interested in such details may now skip directly to Section 4.

For the proof of Proposition 3.5 and subsequent developments, we need to extend the notations and definitions of Section 2 to subsets of an arbitrary job set $K \subseteq N$. For any subsets $J \subseteq K$, let

$$\delta_K^+(J) = \{ij \in \mathcal{A} : i \in J, j \in K \setminus J\} \quad \text{and} \quad \delta_K^-(J) = \{ji \in \mathcal{A} : i \in J, j \in K \setminus J\}.$$

A subset $I \subseteq K$ is *initial in K* if $\delta_K^-(I) = \emptyset$; Similarly, a subset $T \subseteq K$ is *terminal in K* if $\delta_K^+(T) = \emptyset$. Let $\mathcal{I}(K)$ denote the collection of all initial sets in K , and $\mathcal{T}(K)$ that of all terminal sets in K .

Proof of Proposition 3.5: The Proposition vacuously holds for $k = 1$, the case where no decomposition can be found, i.e., the problem is stiff; so, assume $k \geq 2$. For $i = 1, \dots, k$, define $K_i = N \setminus H_{i-1}$. Observe that each K_i is a terminal set in N and each J_i is an initial set in K_i . We prove by induction on i that there exists an optimal schedule for $P(w, p, N, \mathcal{A})$ in which J_i precedes K_i . Since $J_1 = H_1$ is an initial set in $K_1 = N$ which maximizes f_{λ_1} , Theorem 3.1 implies that it is optimal to schedule all jobs in J_1 before all jobs in $K_2 = K_1 \setminus J_1$. Therefore, we may from now on restrict attention to optimal schedules for $P(w, p, K_2, \mathcal{A}_2)$ where \mathcal{A}_2 is the restriction of \mathcal{A} to K_2 . So, by induction, assume we have proved the proposition for all $i \leq h$, and now restrict attention to optimal schedules for $P(w, p, K_h, \mathcal{A}_h)$ where \mathcal{A}_h is the restriction of \mathcal{A} to K_h . We claim that J_h is an initial set in K_h which maximizes f_λ for $\lambda = \lambda_h$.

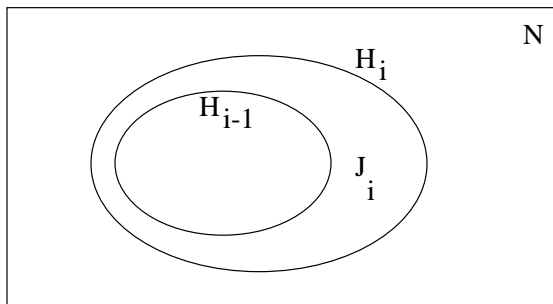


Figure 2: Illustration of Proposition 3.5; $\rho(H_{i-1}) = \lambda_{i-1}$ and $\rho(H_i) = \lambda_i$; $K_i = N - H_{i-1}$ is not shown.

By contradiction, assume that there exists an initial set I in K_h with $f_\lambda(I) > f_\lambda(J_h)$. Since $H_{h-1} \cup I$ is an initial set in N we have

$$\begin{aligned} f_\lambda(H_{h-1}) &\geq f_\lambda(H_{h-1} \cup I) = f_\lambda(I) + f_\lambda(H_{h-1}) \\ &> f_\lambda(H_{h-1}) + f_\lambda(J_h) = f_\lambda(H_h) = f_\lambda(H_{h-1}), \end{aligned}$$

where the last equality follows from H_{h-1} and H_h being both optimal for $\lambda = \lambda_h$. This produces a contradiction. Thus J_h is an initial set in K_h which maximizes f_λ and Theorem 3.1 implies that it is optimal to schedule all jobs in J_h before all jobs in $K_{h+1} = K_h \setminus J_h$, and therefore before J_{h+1} . \square

3.3 Sidney Decompositions and the Semilattice of Sidney Partitions

We now consider in greater detail the structure of the various decompositions one may obtain by using the parametric method described above or Sidney's original decomposition method, as well as through different choices within these methods. We shall start with a few definitions.

For job subset $K \subseteq N$, let

$$\rho^*(K) = \max\{\rho(I) : I \text{ is an initial subset in the subgraph of } G \text{ induced by } K\}.$$

Let (J_1, \dots, J_k) denote an ordered partition of K , that is, a sequence of nonempty subsets of K satisfying $\cup_{i=1}^k J_i = K$ and $J_h \cap J_i = \emptyset$ whenever $h \neq i$. We say that (J_1, \dots, J_k) is a *Sidney*

decomposition of K if, for all $i = 1, \dots, k$, subset J_i is an initial set in $K_i \doteq J_i \cup \dots \cup J_k$ and $\rho(J_i) = \rho^*(K_i)$.

Lemma 3.6 *The GGT algorithm constructs a Sidney decomposition of the job set N .*

Proof: Letting, for $h = 1, \dots, k$, subset J_i be as defined in Proposition 3.5 and, as above, $K_i = N \setminus H_{i-1}$, we only need to prove that $\rho(J_i) = \rho^*(K_i)$. Assume by contradiction that there exists an initial set I in K_i with $\rho(I) > \rho(J_i)$. Note that, as in the proof of Proposition 3.5, we have, $f_{\lambda_i}(J_i) = f_{\lambda_i}(H_{i-1}) - f_{\lambda_i}(H_i) = 0$, so $\lambda_i = \rho(J_i) < \rho(I)$. But then $H_{i-1} \cup I$ is an initial set such that

$$f_{\lambda_i}(H_{i-1} \cup I) = f_{\lambda_i}(H_{i-1}) + f_{\lambda_i}(I) > f_{\lambda_i}(H_{i-1}),$$

a contradiction. Thus we must have $\rho(J_i) = \rho^*(K_i)$. \square

Sidney (1975) constructs a decomposition of the job set N by the following simple process: first, let $K_1 = N$; then for $i = 1, \dots$, and while $K_i \neq \emptyset$, choose a nonempty initial subset J_i of K_i with $\rho(J_i) = \rho^*(K_i)$ and let $K_{i+1} = K_i \setminus J_i$. Clearly, this is a Sidney decomposition as defined above, but it need not coincide with that obtained from the GGT algorithm. We now investigate the collection of all Sidney decompositions of a job set N ; we will show in particular that there is a finest such decomposition and that it can be obtained from the GGT algorithm with modest additional work.

The ρ -profile of a Sidney decomposition $\mathcal{J} = (J_1, \dots, J_l)$ is the sequence $(\lambda_1 > \dots > \lambda_l)$ of the distinct values $\rho(J_i)$ for all members J_i of \mathcal{J} . The proof of the following lemma is immediate, and is left to the interested reader:

Lemma 3.7 *Let $\mathcal{J} = (J_1, \dots, J_l)$ denote a Sidney decomposition of K . If $\rho(J_i) > \rho(J_k)$ then $i < k$.*

A consequence of this Lemma is that in a Sidney decomposition \mathcal{J} , the subsets J_i must be in non increasing order of their ρ values. The *reduction* of a Sidney decomposition \mathcal{J} with

ρ -profile $(\lambda_1 > \dots > \lambda_q)$ is the ordered partition $\mathcal{R} = (R_1, \dots, R_q)$ where, for $i = 1, \dots, q$, $R_i = \bigcup \{J_j : \rho(J_j) = \lambda_i\}$. The proof of the following lemma is immediate, and is also left to the interested reader:

Lemma 3.8 *The reduction of a Sidney decomposition is a Sidney decomposition.*

Theorem 3.9 *All Sidney decompositions of a given instance have the same ρ -profile and the same reduction.*

Proof: Let $\mathcal{J} = (J_1, \dots, J_k)$ and $\mathcal{J}' = (J'_1, \dots, J'_l)$ be two Sidney decompositions of the same job set N , with ρ -profiles $(\lambda_1 > \dots > \lambda_q)$ and $(\lambda'_1 > \dots > \lambda'_r)$, respectively, and reductions $\mathcal{R} = (R_1, \dots, R_q)$ and $\mathcal{R}' = (R'_1, \dots, R'_r)$, respectively. We will show by induction on i that, for all i , $\lambda_i = \lambda'_i$ and $R_i = R'_i$, thus implying that $q = r$ and yielding the result. By definition, we have $\lambda_1 = \rho^*(N) = \lambda'_1$. If $R_1 \neq R'_1$ then let $U = R_1 \cup R'_1$ and assume, wlog, that $R_1 \subset U$. Since U is a union of initial sets in N , it is itself an initial set in N . Therefore, $U \setminus R_1$ is an initial set in $N \setminus R_1$, with $\rho(U \setminus R_1) = \rho^*(N)$ by Lemma 2.1. As $\rho^*(N) > \lambda_2 = \rho^*(N \setminus R_1)$, we get a contradiction. So, we must have $R_1 = R'_1$. The general case proceeds similarly: if $\lambda_h = \lambda'_h$ and $R_h = R'_h$ for all $h \leq i$, then we observe that (R_{i+1}, \dots, R_q) and (R'_{i+1}, \dots, R'_r) are Sidney decomposition of $N \setminus \bigcup_{h \leq i} R_h$ with profiles $(\lambda_{i+1} > \dots > \lambda_q)$ and $(\lambda'_{i+1} > \dots > \lambda'_r)$, respectively; and the same argument applies to show that $\lambda_{i+1} = \lambda'_{i+1}$ and $R_{i+1} = R'_{i+1}$, completing the proof. \square

Accordingly, call the common ρ -profile of all Sidney decompositions of N the ρ -profile of N , and call the common reduction of all Sidney decompositions of N the *reduced Sidney decomposition* of N . Thus all Sidney decompositions of N differ only in the way the subsets R_i in the reduced Sidney decomposition are partitioned and ordered. For our scheduling problem, we will prefer to use a finer decomposition than the reduced Sidney decomposition, whenever one exists; indeed, we prefer to decompose the problem into as many (small) pieces as possible. We will show that this is indeed possible.

A Sidney decomposition $\mathcal{J} = (J_1, \dots, J_k)$ of a job set N defines a partition $\overline{\mathcal{J}} = \{J_1, \dots, J_k\}$ of N consisting of the same subsets, by simply disregarding their order. Call such a partition a *Sidney partition* of N . The subsets J_i are called the *blocks* of the partition. We now characterize all the Sidney decompositions that define a same Sidney partition $\overline{\mathcal{J}}$.

Theorem 3.10 *Let $\overline{\mathcal{J}} = \{J_1, \dots, J_k\}$ be a Sidney partition. An ordered sequence (J_1, \dots, J_k) of its blocks is a Sidney decomposition if and only if, for all distinct $i, j \in \{1, \dots, k\}$,*

- (i). *if $\rho(J_i) > \rho(J_j)$ then $i < j$; and*
- (ii). *if $uv \in \mathcal{A}$ with $u \in J_i$ and $v \in J_j$ then $i < j$.*

Proof: The conditions are clearly necessary: (i) by Lemma 3.7, and (ii) because its violation would imply that J_j is not an initial set in $\bigcup_{h=j}^n J_h$. Therefore, assume that $\mathcal{J} = (J_1, \dots, J_k)$ is an ordered sequence satisfying conditions (i) and (ii), and consider any index $i \in \{1, \dots, k\}$. We prove by induction on i that J_i is a initial set in $K_i = \bigcup_{h=i}^n J_h$ with $\rho(J_i) = \rho^*(K_i)$. This follows immediately from (i) and (ii) when $i = 1$. So assume that this holds for all $h < i$ where $i \geq 2$. This implies that the blocks J_1, \dots, J_{i-1} can be identified by Sidney's algorithm as the first $i-1$ blocks in a Sidney decomposition $\mathcal{J}' = (J_1, \dots, J_{i-1}, J'_i, \dots, J'_r)$. Condition (ii) implies that J_i is an initial set in K_i . Let $(\lambda_1, \dots, \lambda_q)$ denote the ρ -profile of N . Since $\overline{\mathcal{J}}$ is defined by a Sidney decomposition, by Theorem 3.9, $\rho(J_i) = \lambda_j$ for some index j . Therefore, letting \mathcal{R} denote the Sidney reduction of N , we have $J_i \subseteq R_j$. This implies $\rho^*(K_i) \geq \rho(J_i) = \lambda_j$. By condition (i), we have $R_h \cap K_i = \emptyset$ for all $h < j$. By Theorem 3.9 applied to \mathcal{J}' , this implies that $\rho^*(K_i) = \rho(J'_i) \leq \lambda_j$. Therefore $\rho^*(K_i) = \lambda_j = \rho(J_i)$. \square

For our scheduling problem, we will be indifferent between using any of several Sidney decompositions that define a same partition; although the resulting optimal sequences will differ, they will remain (globally) optimal, and the computational effort will be identical, as we will need to solve identical subproblems. We will show that the Sidney partitions of a job set N form a (finite) meet semilattice, and thus that there exists a “finest” Sidney partition.

Recall that the set $\text{Part}(N)$ of all partitions of a set N is a lattice (Grätzer 1978). The associated partial order “ \preceq ” is defined as follows: a partition $\mathcal{Q} = \{Q_1, \dots, Q_q\}$ of N is *finer* than partition $\mathcal{Q}' = \{Q'_1, \dots, Q'_r\}$, denoted $\mathcal{Q} \preceq \mathcal{Q}'$, if for all $i = 1, \dots, q$ there exists an index j such that $Q_i \subseteq Q'_j$. Note that this implies $q \geq r$. A collection $\mathcal{M} \subseteq \text{Part}(N)$ is a *meet sub-semilattice* of $\text{Part}(N)$ if it is closed for the (partition lattice) meet operation, that is, if $\mathcal{Q}, \mathcal{Q}' \in \mathcal{M}$ then their meet $\mathcal{Q} \wedge \mathcal{Q}'$ is also in \mathcal{M} , where $\mathcal{Q} \wedge \mathcal{Q}' \in \mathcal{M}$ is the coarsest partition $\mathcal{Q}'' \in \text{Part}(N)$ such that $\mathcal{Q}'' \preceq \mathcal{Q}$ and $\mathcal{Q}'' \preceq \mathcal{Q}'$.

Theorem 3.11 *The set of all Sidney partitions of a set N is a meet sub-semilattice of the partition lattice $\text{Part}(N)$.*

Proof: Given any two Sidney decompositions $\mathcal{J} = (J_1, \dots, J_k)$ and $\mathcal{J}' = (J'_1, \dots, J'_l)$ of N , we need to prove that the meet (largest lower bound) $\overline{\mathcal{J}} \wedge \overline{\mathcal{J}'}$ of their associated Sidney partitions is also a Sidney partition of N . Recall (e.g., Lemma IV.4.1 in Grätzer 1978) that $\overline{\mathcal{J}} \wedge \overline{\mathcal{J}'}$ is the partition consisting of all the nonempty intersections $H_{ij} = J_i \cap J'_j$. Order these nonempty intersections H_{ij} in lexicographic order of the indices i and j (that is, if H_{ij} is before H_{uv} then either $i < u$ or $i = u$ and $j < v$). It follows that the sequence of all nonempty intersections H_{ij} thus ordered satisfies condition (ii) of Theorem 3.10. By Theorem 3.9, we have $H_{ij} \neq \emptyset$ only if $\rho(J_i) = \rho(J'_j)$. Therefore this ordered sequence of nonempty H_{ij} 's also satisfies condition (i) of Theorem 3.10, implying that $\overline{\mathcal{J}} \wedge \overline{\mathcal{J}'}$ is a Sidney partition. \square

Corollary 3.12 *There exists a finest Sidney decomposition of a set N , unique up to permutations of some of its blocks.*

The following example shows that the set of all Sidney partitions is not closed for the join operation:

Example: let $N = \{a, b, c, d, e\}$ and $\mathcal{A} = \{ab, cd\}$; let all $p_i = w_i = 1$, so all nonempty subsets $J \subseteq N$ have $\rho(J) = \rho^*(N) = 1$; then $\mathcal{J} = (\{a, e\}, \{b, c\}, \{d\})$ and $\mathcal{J}' = (\{a\}, \{b\}, \{c\}, \{d, e\})$ are Sidney decompositions, and yet the join $\overline{\mathcal{J}} \vee \overline{\mathcal{J}'} = \{\{a, d, e\}, \{b, c\}\}$ of their associated Sidney partitions is not a Sidney partition, as neither of its blocks is an initial set in N . \square

Note however that there is a *coarsest*, i.e., least fine, Sidney partition, $\overline{\mathcal{R}}$ defined by the reduced Sidney decomposition \mathcal{R} of N . In contrast with the finest Sidney partition, the coarsest Sidney partition $\overline{\mathcal{R}}$ is unique.

3.4 Constructing a Finest Sidney Decomposition

We now show how to construct a finest Sidney decomposition, using the GGT algorithm described in Section 3.2.

First recall (see, e.g., Hu 1970) that, given a network $\mathcal{N}_\lambda = (N^*, \mathcal{A}^*, c_\lambda)$ with source s and sink t in N^* , the collection of sink sets T of all minimum capacity s, t -cuts forms a sublattice of 2^{N^*} . This sublattice is isomorphic to the sublattice of ideals $I \subseteq N$ with maximum weight $f_\lambda(I) = w(I) - \lambda p(I)$, by simply letting $I = T \setminus \{t\}$, as seen in Section 3.2. Recall also that the largest and smallest such sink sets T_\vee and T_\wedge , respectively, can be obtained by simple labeling. For T_\vee , start from the source s and apply the Ford-Fulkerson labeling procedure: then T_\vee is the set of all unlabeled nodes at the end of the procedure (i.e., all nodes not reachable from the source s in the augmenting network associated with the current maximum flow). For T_\wedge , take all nodes from which one can reach the sink t in the augmenting network (see Hu (1970) for details). Note that each procedure requires $O(m + n)$ operations, where $n = |N| = |N^*| - 2$ and $m = |\mathcal{A}|$ (so the number of arcs in \mathcal{N}_λ is $m + 2n$).

Since, by Theorem 3.9, $R_1 = T_\vee \setminus \{t\}$ defines the largest ideal with maximum weight $f_\lambda(I)$, it follows that we immediately obtain the reduced Sidney decomposition by using Ford-Fulkerson's original labeling procedure at each step of the GGT algorithm.

For our scheduling purposes, however, we want to construct a finest Sidney decomposition. Recall that the GGT algorithm identifies a sequence of breakpoints $+\infty = \lambda_0 > \lambda_1 > \dots > \lambda_k \geq \lambda_{k+1} = 0$, where, as follows from Sections 3.2 and 3.3, $(\lambda_1, \dots, \lambda_k)$ is the ρ -profile of N . Letting, as above, $\mathcal{R} = (R_1, \dots, R_k)$ denote the reduced Sidney decomposition of N , each set $I_i = \bigcup_{h \leq i} R_h$ is, for $i = 0, \dots, n$, the largest ideal I in N with maximum weight $f_{\lambda_i}(I)$. Consider a step of the GGT algorithm where we have determined a maximum flow for $\lambda = \lambda_i$. We now

seek to determine a finest Sidney decomposition, which amounts to finding a maximal nested family of minimum s, t -cuts in \mathcal{N}_λ . All minimum s, t -cuts in \mathcal{N}_λ can be obtained as follows (details can be found in Picard and Queyranne (1980):

Let $AG(\mathcal{N}_\lambda, f)$ be the augmentation graph associated with the optimal flow f on \mathcal{N}_λ and let A_1, \dots, A_u be its strongly connected components. Let AG' be the directed graph obtained from AG by contracting to a single vertex a_j the component A_j for $j = 1, \dots, u$. Observe that AG' is a directed acyclic graph, thus inducing a natural partial order on its nodes. Any minimum s, t -cut in \mathcal{N}_λ is the set of arcs entering the union of components corresponding to an initial set in AG' . A maximal nested family of such cuts can be obtained by ordering the nodes of AG' according to any order compatible with the partial order represented by AG' , say $\{b_1, \dots, b_u\}$ and considering the cuts generated by the sets $b_1 \cup \dots \cup b_j$ for $j = 1, \dots, u - 1$.

As all the above operations can be carried out in time $O(m + n)$, we have shown:

Theorem 3.13 *Let $MF(v, a)$ denote the time required to solve a maximum flow problem in a network with v nodes and a arcs, using an algorithm compatible with the Gallo-Grigoriadis-Tarjan parametric maximum flow approach (Gallo, Grigoriadis and Tarjan 1989). Then a finest Sidney decomposition of a scheduling problem with n jobs and m precedence constraints can be constructed in $MF(n, m + 2n) + O(n(m + n))$ time. The coarsest Sidney decomposition can be constructed in $MF(n, m + 2n) + O((m + n))$ time.*

4 Properties of the LP Relaxation: tight sets and decompositions

In this section, we study an LP relaxation of the problem. We explore some interesting properties of the LP optimal solutions. We derive a necessary and sufficient condition for the Sidney decomposition of the problem. Finally, we show that any feasible solution of a stiff instance has objective value at most twice the optimal value. (This has been proved independently by Chekuri and Motwani 1999.)

For convenience, we introduce some notation. For any vectors $u, v \in R^N$ and $J \subseteq N$, let

$u(J) = \sum_{j \in J} u_j$, $u^2(J) = \sum_{j \in J} u_j^2$, and $u * v(J) = \sum_{j \in J} u_j v_j$. The following set function plays a fundamental role in the study of scheduling polyhedra (see, e.g., Queyranne 1993 or Queyranne and Schulz 1996): for $J \subseteq N$, let

$$g(J) = \frac{1}{2} [p(J)^2 + p^2(J)].$$

The following identity (Queyranne 1993) will be useful: for any $J, H \subseteq N$,

$$(2) \quad g(J \cup H) + g(J \cap H) = g(J) + g(H) + p(J \setminus H)p(H \setminus J).$$

For any $J \subseteq N$, inequality $p * C(J) \geq g(J)$ is known to be valid for (1), and it is called a *parallel inequality* (Queyranne and Wang 1991a)

Without loss of generality we may assume that the precedence graph G is *transitively reduced*, that is, \mathcal{A} is the minimum collection of arcs representing the partial order. (This is also called the Hasse diagram of the partial order).

Let A denote the $|\mathcal{A}| \times |N|$ *arc-job* incidence matrix of G , where rows of A are indexed by arcs $ij \in \mathcal{A}$, columns by jobs $h \in N$, and

$$A_{ij,h} = \begin{cases} -1 & \text{if } h = i \text{ and } ij \in \mathcal{A} \\ 1 & \text{if } h = j \text{ and } ij \in \mathcal{A} \\ 0 & \text{otherwise} \end{cases}$$

Let A^h denote the column of A indexed by job h . Also define the column vector $b \in R^{\mathcal{A}}$ as $b_{ij} = p_j$ for all $ij \in \mathcal{A}$. Let $\mathbf{0}$ denote a vector of all 0's, and $\mathbf{1}$ a column vector of all 1's.

Replacing the disjunctive constraints and $C \geq p$ in (1) with all parallel inequalities, we obtain the following LP relaxation.

$$(3) \quad \begin{aligned} wC^* &= \min \quad wC \\ \text{s.t.} \quad & p * C(J) \geq g(J), \quad \forall J \subseteq N \\ & AC \geq b. \end{aligned}$$

Using a variable μ_J for each $J \subseteq N$ and a variable y_{ij} for each arc $ij \in \mathcal{A}$, the LP dual formulation of (3) is

$$(4) \quad \begin{aligned} D(\mu^*, y^*) &= \max \sum (g(J)\mu_J : J \subseteq N) + yb \\ \text{s.t.} \quad &\sum (p_h \mu_J : J \ni h, J \subseteq N) + yA^h = w_h, \quad \forall h \in N \\ &\mu \geq \mathbf{0}, \quad y \geq \mathbf{0}, \end{aligned}$$

where (μ^*, y^*) denote an optimal dual solution.

Now, for any optimal solution C to (3), define

$$\tau(C) = \{J : J \subseteq N, p * C(J) = g(J)\}.$$

That is, $\tau(C)$ is a family of *tight* sets. By convention, $\emptyset \in \tau(C)$.

Lemma 4.1 *Let $J \in \tau(C)$ be any nontrivial tight set. Then*

$$(5) \quad \begin{aligned} (i) \quad &\forall i \in J, \quad C_i \leq p(J) \quad \text{and it holds with equality iff } J \setminus \{i\} \in \tau(C); \\ (ii) \quad &\forall j \notin J, \quad C_j \geq p(J) + p_j \quad \text{and it holds with equality iff } J \cup \{j\} \in \tau(C). \end{aligned}$$

Proof: Note that by (2), $g(J) - g(J \setminus \{i\}) = p_i p(J)$. Thus, (i) follows from

$$g(J) = p_i C_i + p * C(J \setminus \{i\}) \geq p_i C_i + g(J \setminus \{i\}).$$

Similarly, since $g(J \cup \{j\}) - g(J) = p_j(p(J) + p_j)$, (ii) follows from

$$g(J \cup \{j\}) \leq p_j C_j + p * C(J) = p_j C_j + g(J).$$

Note that we have used the fact that all parallel inequalities are valid. □

Lemma 4.2 *Let $J \in \tau(C)$ be any tight subset. If one of the following conditions holds:*

- (i). *there exists no tight set \hat{J} with $\hat{J} \subset J$ and $|\hat{J}| = |J| - 1$,*
- (ii). *there exists no tight set \hat{J} with $J \subset \hat{J}$ and $|\hat{J}| = |J| + 1$,*

then $C_j - C_i > p_j$, for all $i \in J$ and $j \in N \setminus J$.

Proof: Suppose that for for some $i \in J$ and $j \in N \setminus J$, $C_j - C_i \leq p_j$. Then by (i) and (ii) of (5) we have $C_j - p(J) \leq C_j - C_i \leq p_j \leq C_j - p(J)$. This implies

(a) $C_i = p(J)$ and (b) $C_j = p(J) + p_j$.

(1) If $|J| \geq 1$ and $C_j - C_i \leq p_j$ for some $i \in J$ and $j \in N \setminus J$ then, by (a) above and (i) of (5), $J \setminus \{i\} \in \tau(C)$, a contradiction to the minimality of J .

(2) Observe that (b) above and (ii) of (5) imply that $\hat{J} \cup \{j\} \in \tau(C)$ with $|\hat{J}| = |J| + 1$, again a contradiction. \square

Note that it is not necessary that \mathcal{A} contain the pair ij or ji .

The following proposition establishes some important properties of tight sets.

Proposition 4.3 *For any optimal solution C to (3),*

(i). $\tau(C)$ contains a nonempty set;

(ii). for any pair $J, H \in \tau(C)$, either $J \subset H$ or $H \subset J$ holds;

(iii). each $J \in \tau(C)$ is initial;

(iv). $N \in \tau(C)$ if w is not tailing off.

Proof: (i): If (i) does not hold, then by complementary slackness, $\mu^* = \mathbf{0}$. So $w = y^* A$, and $w(N) = w\mathbf{1} = y^* A\mathbf{1} = 0$, implying $w = \mathbf{0}$, a contradiction.

(ii): If (ii) is false, then there exists a pair $J, H \in \tau(C)$ with $p(J \setminus H)p(H \setminus J) > 0$. Using two parallel inequalities induced by $J \cup H$ and $J \cap H$ to obtain the first inequality below and the identity (2) for the second line, we obtain a contradiction since

$$\begin{aligned} g(J \cup H) + g(J \cap H) &\leq p * C(J \cup H) + p * C(J \cap H) = p * C(J) + p * C(H) \\ &= g(J) + g(H) = g(J \cup H) + g(J \cap H) - p(J \setminus H)p(H \setminus J). \end{aligned}$$

(iii): If some $J \in \tau(C)$ is not initial, then there exists some $j \notin J$ but $ji \in \mathcal{A}$ for some $i \in J$. Then $C_j \leq C_i - p_i$, and $C_i \leq p(J)$ by (i) of (5). This implies that $C_j < p(J)$, a contradiction to inequality (ii) of (5).

(iv) Suppose that $N \notin \tau(C)$. Then by (i) and (ii), there exists a unique maximal tight set J^* with $J^* \neq N$. Let $H = N \setminus J^*$, and note that H is terminal since J^* is initial by (iii). Define \hat{C} by $\hat{C}_h = C_h$ if $h \in J^*$ and $\hat{C}_h = C_h - \epsilon$ if $h \in H$. By maximality of J^* and (ii) of Lemma 4.2, we have $C_j - C_i > p_j$ for all $ij \in \delta^+(J^*)$. Furthermore, the maximality of J^* ensures that no tight set contains a node in H . Therefore, for sufficiently small $\epsilon > 0$, \hat{C} is primal feasible. But $w\hat{C} \leq w\hat{C} = wC - \epsilon w(H)$, implying $w_j = 0$ for all $j \in H$ and w is tailing off. \square

Theorem 4.4 *Assume that $G = (N, \mathcal{A})$ does not contain a directed Hamiltonian path. The problem $P(w, p, N, \mathcal{A})$ is Sidney decomposable if and only if there exists an optimal LP solution C such that $p * C(J) = g(J)$ holds for some nontrivial subset $J \subset N$.*

Proof: *Necessity:* Let $I \in \mathcal{I}$ be a nontrivial initial set such that $\rho(I) = w(I)/p(I) = \rho^*$. Let $H = N \setminus I$. Since $P(w, p, N, \mathcal{A})$ is Sidney decomposable and using Lemma 2.1, we have

$$(6) \quad \frac{w(I)}{p(I)} \geq \frac{w(N)}{p(N)} = \frac{w(I) + w(H)}{p(I) + p(H)} \iff \frac{w(I)}{p(I)} \geq \frac{w(H)}{p(H)}.$$

Now, consider any optimal LP solution C . If there exists some nontrivial subset $J \subset N$ with $p * C(J) = g(J)$, we are done. Otherwise, for $\epsilon > 0$, define $C_i^\epsilon = C_i - \epsilon p(H)$ for all $i \in I$ and $C_j^\epsilon = C_j + \epsilon p(I)$ for all $j \in H$. Observe that for any $\epsilon > 0$, the following inequality

$$(7) \quad p * C^\epsilon(N) = p * C(I) - \epsilon p(H)p(I) + p * C(H) + \epsilon p(I)p(H) \geq g(N)$$

and all precedence constraints are still satisfied by C^ϵ (since I is initial). Furthermore, by (6)

$$wC \leq wC^\epsilon = wC - \epsilon w(I)p(H) + \epsilon w(H)p(I) \leq wC.$$

So C^ϵ is also an optimal solution if no parallel inequality $p * C^\epsilon(J) \geq g(J)$ induced by nontrivial subset $J \subset N$ is violated as ϵ increases. Clearly, we can increase ϵ until $p * C^\epsilon(J^\epsilon) \geq g(J^\epsilon)$ becomes binding for some nontrivial subset J^ϵ . The resulting C^ϵ is the required solution.

Sufficiency: Let C be the optimal solution with $\tau(C)$ containing some nontrivial subset. We need to find some nontrivial initial subset J^* with $\rho(J^*) \geq \rho(N)$.

First, observe that if $N \notin \tau(C)$, then by (iv) of Proposition 4.3, w is tailing off, which implies that there exists some nontrivial terminal subset H with $w(H) = 0$. So $J^* = N \setminus H$ is the required initial set, and we are done. Thus, for the rest of the proof assume that $|\tau(C)| \geq 3$ (since $\tau(C)$ contains the empty set, a nontrivial tight set and N).

By (ii) and (iii) of Proposition 4.3, we know that $\tau(C)$ contains k nested nonempty tight sets $J_1 \subset \dots \subset J_k$, where $2 \leq k \leq n$ and that all these sets are initial. We distinguish the following two cases.

CASE 1: $2 \leq k < n$.

(I). We first claim that, for some q with $1 \leq q < n$, $C_j - C_i > p_j$ for all $ij \in \delta^+(J_q)$.

If $|J_1| \geq 2$, then (i) of Lemma 4.2 implies that the claim holds; Otherwise, since $k < n$, there exist tight sets J_q and J_{q+1} with $|J_{q+1}| \geq |J_q| + 2$. Then the claim follows from (ii) of Lemma 4.2. This proves the claim.

(II). Let $I = J_q$ and $T = N \setminus J_q$. Define $C_i^\epsilon = C_i + \epsilon p(T)$ for all $i \in I$ and $C_j^\epsilon = C_j - \epsilon p(I)$ for all $j \in T$. For sufficiently small $\epsilon > 0$, C^ϵ violates no precedence constraints by the above claim, and moreover, using 4.3 (ii), it is straightforward to verify that

$$p * C^\epsilon(Q) > g(Q), \text{ for all } Q \neq N, \text{ and that } p * C^\epsilon(N) = g(N).$$

So C^ϵ is a feasible solution, and

$$0 \leq wC^\epsilon - wC = \epsilon w(I)p(T) - \epsilon w(T)p(I) \implies \frac{w(I)}{p(I)} \geq \frac{w(T)}{p(T)}.$$

It follows that $\rho(I) \geq \rho(N)$ as required.

CASE 2: $k = n$. Without loss of generality assume that $C_1 < C_2 < \dots < C_n$. It follows from repeated application of Lemma 4.1 that C forms a schedule, that is, $C_1 = p_1$, $C_{j+1} = C_j + p_j$ for $j = 1, \dots, n-1$. First observe that $C_j - C_i > p_j$ for any $ij \in \mathcal{A}$ with $j \geq i+2$ (Otherwise, by Lemma 4.1, $J_i \cup \{j\} \notin \tau(C)$ is also tight, a contradiction.) If there exists some proper tight subset J_q such that $(q, q+1) \notin \mathcal{A}$, then using the above observation and the construction of C^ϵ

as in (II) of CASE 1, we show that $I = J_q$ is the required subset and we are done. Otherwise, we must have $(q, q + 1) \in \mathcal{A}$ for $q = 1, \dots, n - 1$. So \mathcal{A} forms a chain, again a contradiction. \square

When $P(w, p, N, \mathcal{A})$ is stiff (i.e., not Sidney decomposable), some interesting consequences follow from the above theorem. First, one obtains a new (trivially computable) lower bound for the optimal objective value z^* . Second, any feasible schedule with no idle times has its objective value within a factor of 2 of the optimal objective value.

Theorem 4.5 *If $P(w, p, N, \mathcal{A})$ is stiff, then the optimal LP value $wC^* \geq w(N)p(N)/2$. Moreover, there exists a family of instances such that this inequality is asymptotically tight.*

Proof: Since $P(w, p, N, \mathcal{A})$ is stiff, by Theorem 4.4, $\tau(C) = \{N\}$ for any optimal LP solution C (since $w \neq \mathbf{0}$.) By duality, $w = \mu_N^* p + y^* A$, and the optimal LP value is $wC^* = \mu_N^* g(N) + y^* b$. Since $w(N)p(N) = w\mathbf{1}p(N) = (\mu_N^* p(N) + 0)p(N) = \mu_N^* p(N)^2$,

$$wC^* = \mu_N^* g(N) + y^* b \geq \mu_N^* \frac{p(N)^2}{2} = \frac{w(N)p(N)}{2},$$

as required. The required family of instances can be constructed as follows. Let $N = \{1, \dots, n\}$ and $\mathcal{A} = \{jn : j = 1, \dots, n - 1\}$. Let $w_j = 0$ for $j = 1, \dots, n - 1$ and $w_n = n$, and let $p_j = 1$, for $j = 1, \dots, n$. The optimal LP solution C^* is given by $C_j^* = \frac{n^2+n-2}{2n}$ for $j = 1, \dots, n - 1$ and $C_n^* = \frac{n^2+3n-2}{2n}$ with $wC^* = \frac{n^2+3n-2}{2n}$ and $\frac{w(N)p(N)}{2} = \frac{n}{2}$. \square

Corollary 4.6 *If $P(w, p, N, \mathcal{A})$ is stiff, then any feasible schedule with no idle times has its objective value within a factor of 2 of the optimal value.*

Using the same LP formulation, Schulz (1996) and Hall et al. (1997) obtained 2-approximation algorithms by constructing a feasible solution with value at most twice the value of the linear relaxation. Corollary 4.6 has the same flavor, but shows that, for a stiff instance, even the worst feasible solution is within a factor of 2 of the optimal value.

5 Solving the LP for a Stiff Instance

In this section, we show how to solve the LP (3) for a stiff instance $P(w, p, N, \mathcal{A})$ by network flow techniques. By Theorem 4.4, the LP (3) for a stiff instance reduces to the inequalities generated by the precedence constraints and to the equality $p * C(N) = g(N)$. In the remainder of this section, we find convenient to work with variables associated with starting times instead of completion times. For $J \subseteq N$, we define $g'(J) = \frac{1}{2} [p(J)^2 - p^2(J)]$ and for $j \in N$, we use $S_j = C_j - p_j$ to denote the start time for job j . Hence

$$p_j S_j = p_j C_j - p_j^2, \quad \text{and} \quad C_j - C_i \geq p_j \iff S_j - S_i \geq p_i.$$

Rewriting the LP (3) with these variables (see Queyranne and Schulz 1996), we obtain:

$$(8) \quad \begin{aligned} \min \quad & z = wS \\ \text{s.t.} \quad & p S = g'(N) \\ & S_j - S_i \geq p_i \quad \text{for all } ij \in \mathcal{A} \\ & S \geq \mathbf{0}. \end{aligned}$$

Let $\rho = w(N)/p(N)$ and $\bar{w} = w - \rho p$. Consider the Lagrangian relaxation of this problem obtained by dualizing the equality constraint with multiplier ρ , i.e.:

$$(9) \quad \begin{aligned} \min \quad & z' = wS - \rho (p S - g'(N)) = \bar{w} S + \rho g'(N) \\ \text{s.t.} \quad & S_j - S_i \geq p_i \quad \text{for all } ij \in \mathcal{A} \\ & S \geq \mathbf{0}. \end{aligned}$$

We say that an arc $ij \in \mathcal{A}$ is *tight* for a solution S if $S_j - S_i = p_i$. The next Lemma implies that $z = z'$.

Lemma 5.1 *There exists a solution S^* optimal for both LP (8) and LP (9).*

Proof: Since the problem $P(w, p, N, \mathcal{A})$ is stiff, for each initial set $I \in \mathcal{I}$ we have $\bar{w}(I) = w(I) - \frac{w(N)}{p(N)}p(I) < 0$ and thus for each terminal set $T \in \mathcal{T}$ we have $\bar{w}(T) > 0$. Since the

extreme rays of LP (9) are generated by the characteristic vectors of the terminal sets, the optimal value of LP (9) is bounded. As LP (9) is contained in the positive orthant, there exists an optimal extreme point S . Let K be a connected component of the subgraph of G induced by the tight arcs for S . Note that K contains a node i with $S_i = 0$, for otherwise, adding (resp. subtracting) a small $\epsilon > 0$ to S_j for all j in K would yield a feasible point S^+ (resp. S^-). But then $S = (S^+ + S^-)/2$, a contradiction. Hence, for each node $j \in N$ there exists a path of tight arcs joining j to a node $i(j)$ with $S_{i(j)} = 0$ (ignoring the orientation of the arcs).

Let $t = \arg \max \{S_i \mid i \in N\}$ and t_0, \dots, t_k, t be a simple path of tight arcs joining t_0 to t with $S_{t_0} = 0$. Then we have $S_t \leq p_{t_0} + \dots + p_{t_k}$ and, by choice of t , $S_{t'} \leq S_t$ for all $t' \in N$. Consider now the schedule obtained by starting with the jobs t_0, \dots, t_k, t in that order and then putting the remaining jobs in arbitrary order. (This schedule is probably not feasible for LP (9), but this is of no importance). Denote by y the starting times of that schedule. Note that $S \leq y$ and that $p y = g'(N)$. It follows that $p S \leq g'(N)$ and thus, for $q = (g'(N) - p S)/p(N)$, we have $q \geq 0$. Hence $S^* = (S + q \mathbf{1})$ is feasible for LP (9) and satisfies $p S^* = g'(N)$, i.e. S^* is feasible for LP (8). Moreover, since $\bar{w}(N) = 0$, we have $\bar{w} S^* = \bar{w} S$, implying that S^* is an optimum solution to LP (9) feasible for LP (8), i.e. an optimum solution to LP (8). \square

The LP (9) is, in fact, a problem on node potentials with non-negativity constraints. Hence, if we forget about the non-negativity constraints, it is the dual of a max-cost flow problem with supply-demand vector \bar{w} and with arc capacities all infinity. By solving this max-cost flow problem and constructing a dual feasible solution by complementary slackness, we can find an optimal solution of LP (8), as outlined in the proof of the previous lemma. The complexity of the whole procedure is dominated by the time to compute a max-cost flow, e.g. $O((|\mathcal{A}| \log n)(|\mathcal{A}| + n \log n))$ (Ahuja et al. 1993).

Depending on the algorithm chosen for solving the max-cost flow problem, the dual variables may be readily available. Otherwise, starting from an optimal solution f^* of the flow problem, we consider the connected components K_1, \dots, K_u induced by the arcs ij with $f_{ij}^* > 0$. For each $1 \leq v \leq u$, one can set potential $S_i \geq 0$ for the nodes $i \in K_v$ such that all arcs of K_v are

tight for S and each component contains a node with potential 0. This solution S may violate some precedence constraints associated with arc $ij \in \mathcal{A}$ with i and j in different connected components. But since the potential associated to the nodes of one connected component are defined up to an additive constant, it remains to determine additive constants $\Delta_1, \dots, \Delta_u \geq 0$ such that adding Δ_v to the potential of the nodes in K_v for all $1 \leq v \leq u$ yield a feasible optimal solution to LP (9) .

These Δ 's can be found by solving a longest path problem the digraph G' obtained from G by contracting each connected component to a single node, and replacing arc ij joining i in component K_u with j in component K_v by an arc joining K_u to K_v with weight $S_i + p_i - S_j$. Add one node s joined to all nodes in G' by an arc of length 0. Since LP (9) is feasible, G' does not contain a directed cycle with positive weight and we can use Bellman-Ford algorithm to compute the longest path from s to the other nodes in G' in $O(nm)$ operations. It is straightforward to check that using the value of the longest path from s to the node K_u for Δ_u yield a feasible solution to LP (9) that satisfies the complementary slackness condition with f^* , implying the optimality of this solution.

6 Computational results

We implemented the algorithms described in Sections 3, 4 and 5 and tested its performance on random instances. We first give a high level description of the algorithm before detailing its steps and explicating some of the terms it uses in the remainder of the section.

1. Initialize the ordered list L of subproblems as containing only the initial problem;
2. For each subproblem P in the list L , repeat
 - 2.1. if P is series decomposable, replace P in L by the ordered subproblems of its series decomposition; break;
 - 2.2. If P is Sidney decomposable, replace P in L by the ordered subproblems of a finest Sidney decomposition of P ; break;

/* Now P is non series decomposable and stiff */

- 2.3. Remove P from L ;
 - 2.4. Compute the value of the LP (3) for P ; Try to improve the value of this lower bound;
 - 2.5. Compute a heuristic solution for P ;
3. Paste the heuristic solutions and LP values of the non decomposable subproblems to obtain an upper bound and lower bound on the optimal value for the initial problem.

Decompositions: Steps 2.1 and 2.2. We use two types of decompositions: series decompositions and Sidney decompositions. Series decompositions occur when there exists an initial set I such that each node in $N \setminus I$ may be reached from each node in I by a directed path in G . In other words, the jobs in I have to be completed before any job in $N \setminus I$ can start. Thus, jobs in I appear before jobs in $N \setminus I$ in any feasible schedule. Series decompositions can be found in $O(n + m)$ for a graph with n nodes and m arcs. The algorithm to find the Sidney decompositions outlined in Section 3 uses a parametric network flow algorithm. Unfortunately, we could not find an implementation of the the parametric network flow algorithm of Gallo, Grigoriadis and Tarjan and thus we find the Sidney decompositions by $O(n)$ calls to a max-flow code (we use the code *Maxflow* written by Goldfarb and Grigoriadis 1988).

Solving the LP: Step 2.4. To find the optimal solution of the LP (3) for a stiff instance, we follow the discussion of Section 5 to formulate the problem in the form of LP (9). We solve its dual, a max-cost flow problem, with the network simplex solver of the linear optimizer CPLEX4.0. We found that, for our problems, it is at least as efficient as the max-cost flow codes at our disposal. From the optimal flow and as discussed in Section 5, we construct an optimal solution of LP (9) from which an optimal solution of the LP (8) and then of the LP (3) are easily obtained. The optimum value of the LP (3) is denoted by LP .

Lower Bound Improvement: Improving the lower bound given by the LP (9) is possible, as observed by Hoogeveen and Van de Velde (1996). They give a strengthening for the Lagrangian relaxation of the precedence constraints of the LP (9). We did not implement an

ascent method to find the best Lagrange multipliers as suggested by Hoogeveen and Van de Velde, but merely used the max-cost flow solution. The value of this improved lower bound is denoted by *LPL*.

Heuristics: Step 2.5. A simple heuristic to find a relatively good solution of the scheduling problem is to order the jobs as indicated by the optimal solution C of the LP (3), starting with the job with the smallest entry in C . This ordering is feasible since if $ij \in \mathcal{A}$ then $C_j - C_i \geq p_j$ is a constraint of the LP and thus $C_i < C_j$. The solution obtained in this way is denoted by *OUB* for Original Upper Bound.

Note that it is possible to replace C by a point C' obtained by subtracting $\alpha_i p_i$ to C_i for all $i \in N$, where α_i is any number in $[0, 1)$. Ordering the jobs according to C' will also give a feasible solution. This idea, introduced by Phillips et al. (1998) for converting preemptive schedules to non preemptive ones for problems with release dates, is part of several approximation algorithms for scheduling problem (Goemans et al. 2002). We generated 10 such feasible schedule O_1, \dots, O_{10} for each subproblem by randomizing the α_i 's.

Most of the time, substantial local improvements of a feasible solution are possible by permuting two consecutive jobs, or more generally, by permuting two adjacent groups of at most d jobs, for a fixed d . Applying this improvement heuristic to several feasible schedules usually yields a very good feasible solution. However, when the number of jobs increases above a few hundred, this procedure may become quite time consuming, depending on the value of d (we used $d = 10$). In the results reported below, the time spent for this heuristic may be as high as 50% of the total time for problems with 1,000 jobs or more. Obviously, it is possible to devise a much faster heuristic (using a smaller value for d , or looking for improvements only for a fixed amount of time) that would return almost the same results. But since we use the best known feasible solution to approximate the gap, and we want this gap to be as small as possible, we don't mind spending a large fraction of the time in the heuristic procedure. We denote by *UB* the best solution found by this improvement procedure on the 10 feasible schedules O_1, \dots, O_{10} .

Pasting the solutions: Step 3. In Step 2 of the algorithm, we construct an ordered list of subproblems (J_1, \dots, J_k) forming a partition of N and for each of these subproblems (also called piece) J_r , we find an heuristic solution UB_r and compute the lower bound LP_r corresponding to the LP (3). Pasting the solutions of the subproblems together, we get an upper bound UB and a lower bound LP on the value of the optimal solution of the original problem using:

$$UB = \sum_{i=1}^k [UB_i + w(J_i) \sum_{j=1}^{i-1} p(J_j)] \quad \text{and} \quad LP = \sum_{i=1}^k [LP_i + w(J_i) \sum_{j=1}^{i-1} p(J_j)].$$

We also compute the simple lower bound WP derived from Theorem 4.4, i.e.:

$$WP = \sum_{i=1}^k [w(J_i)p(J_i)/2 + w(J_i) \sum_{j=1}^{i-1} p(J_j)].$$

Instances: Problems are generated as proposed by Potts (1985). The precedence graph is a random acyclic directed graph on n nodes with density π , i.e. a directed graph obtained by selecting independently and uniformly arc ij , with $i < j$, with probability π . For a given value of n , 20 problems are generated, 2 for each value of

$$\pi \in \{0.001, 0.02, 0.04, 0.06, 0.08, 0.1, 0.15, 0.2, 0.3, 0.5\}.$$

The processing times are drawn from the discrete uniform distribution on $[1, 100]$ and the weights are drawn from the discrete uniform distribution on $[1, 10]$. The graphs are then replaced by their transitive reduction. The number of arcs in the resulting digraphs varies, of course, with π . For example, for problems with 1000 jobs, we get roughly 430 arcs for $\pi = 0.001$, 4000 arcs for $\pi = 0.02$, 3000 arcs for $\pi = 0.08$, and 2000 arcs for $\pi = 0.3$,

We report results for problems generated in this way for 100, 500, 1000, 1500 and 2000 jobs. We partition the 20 problems generated for a value of n into four groups: problems 1-2 ($\pi = 0.001$), problems 3-10 ($\pi \in \{0.02, 0.04, 0.06, 0.08\}$), problems 11-18 ($\pi \in \{0.1, 0.15, 0.2, 0.3\}$) and problems 19-20 ($\pi = 0.5$). The problems generated with $\pi = 0.001$ (problems 1 and 2) or $\pi = 0.5$ (problems 19 and 20) are usually easy instances, since the former are close to a digraph with no arcs and the latter resemble a complete acyclic digraph. The other problems are harder but of relatively homogeneous difficulty inside a group.

Results: To assess the efficiency of the local improvement heuristic and of the Lagrangian relaxation procedure, we report the percentage of the gap that is closed by the corresponding procedure, *gapim* for the improvement heuristic and *gapl* for the Lagrangian procedure. Ideally, if the optimal value of a problem is *OS*, the best we can hope is that the local improvement heuristic close the gap between *OUB* and *OS*. Unfortunately, for most of the problems we consider, the value of the optimal solution is not known. Hence we use *LPL* instead of *OS* to obtain a pessimistic estimator of the performance of the heuristic. Similarly, the Lagrangian procedure may be able to close the gap between *LP* and *OS*, and we use *UB* instead of *OS* to get an underestimation of the performance of the procedure. More precisely, we report

$$gapim = 100 \frac{OUB - UB}{OUB - LPL} \quad \text{and} \quad gapl = 100 \frac{LPL - LP}{UB - LP}.$$

In addition, we also report the *gap* between the best heuristic solution *UB* and the best lower bound *LPL* and the ratio *r_wp* between the lower bounds *WP* and *LP*:

$$gap = 100 \frac{UB - LPL}{LPL} \quad \text{and} \quad r_wp = 100 \frac{WP}{LP}.$$

For a group of instances, we report the average value of *gap*, *gapim*, *gapl*, *r_wp*, the size of the largest piece in the decomposition *mpc* and the cpu time in seconds *cpu* with names prefixed with "a.". We also report the maximum of *gap*, *mpc* and *cpu* with names prefixed with "m.". The machine used was a Sun Sparcstation 5 with a 400Mz processor running SunOS5 and using the compiler gcc2.6.3.

Several conclusions may be drawn from Table 1. First, the gap between the best known feasible solution *UB* and the lower bound *LPL* is quite small: the maximum value of *gap* is 3.48 %, attained for a single problem with 1000 jobs. The average gap is between 1% and 1.5%. Interestingly, the gap seems to decrease as the number of jobs increases above 1000. The local improvement procedure for a feasible solution closes in average 20-40 % of the gap between *OUB* and *LPL*, whereas the Lagrangian relaxation closes in average 3-10 % of the gap between *UB* and *LP*. Here, it seems that the latter procedure is more efficient for small values of *n*,

closing up to 31 % of the gap between the LP solution and the best known feasible solution for problems with 100 jobs. We also note that the simple bound derived from Theorem 4.4 is a very good bound, in particular for problems with 1500 jobs or more, considering the simplicity of its computation. The decompositions are able to reduce in average the size of the largest piece to, roughly, 60% of the initial size. The cpu times are here as an indication that the decomposition procedure does not take a huge amount of time. Interestingly enough, the time needed to apply our randomized heuristic on a problem with ≥ 500 jobs (i.e. no decomposition is performed and 10 calls to the heuristic are made) takes far more time than the time needed to decompose the problem, apply the randomized heuristic on the pieces of the decomposition and paste the solutions. The value of the solutions obtained are also slightly better when decompositions are in use.

Note that the LP formulation we use on a stiff piece is one of the simplest formulations, i.e. it reduces to the precedence constraints and the parallel inequality on the jobs in the piece. This formulation is generally very weak for random instances, but our results show that it is quite strong for the non-series decomposable stiff instances we get by decompositions.

Acknowledgments

We wish to thank two anonymous referees for their helpful comments and suggestions.

7 References

Ahuja R.K., Magnanti T.L., Orlin J.B. 1993. *Network Flows*, Prentice Hall.Englewoods Cliff, NJ.

Balas E. 1985. “On the Facial Structure of Scheduling Polyhedra”, *Mathematical Programming Study* 24, 179–218.

Chekuri C., Motwani R. 1999. “Precedence Constrained Scheduling to Minimize Sum of Weighted Completion Times on a Single Machine”, *Discrete Applied Mathematics* 98, 29–38.

- Davey B.A., Priestley H.A. 1990. *Introduction to Lattices and Order*, Cambridge University Press, Cambridge.
- Gallo G., Grigoriadis M.D., Tarjan R.E. 1989. "A Fast Parametric Maximum Flow Algorithm and Applications", *SIAM J. Computing* 18, 30–55.
- Goemans M.X., Queyranne M., Schulz A.S., Skutella M., Wang Y. 2002. "Single Machine Scheduling with Release Dates", *SIAM Journal on Discrete Mathematics* 15, 165–192.
- Grätzer G. 1978. *General Lattice Theory*, Academic Press, New York.
- Goldfarb D., Grigoriadis M.D. 1988. "A Computational Comparison of the Dinic and Network Simplex Methods for Maximum Flow", *Annals of Operations Research* 13, 83–123.
- Hu T.C. 1970. *Integer Programming and Network Flows*, Addison–Wesley, Reading, Massachusetts.
- Hall L.A., Schulz A.S., Shmoys D.B., Wein J. 1997. "Scheduling to Minimize Average Completion Time: Off–Line and On–Line Approximation Algorithms", *Mathematics of Operations Research* 22, 513–544.
- Hoogeveen J.A., Van de Velde S.L. 1996. "Stronger Lagrangian Bounds by Use of Slack Variables: Applications to Machine Scheduling Problems", *Mathematical Programming* 70, 173–190.
- Lawler E.L. 1976. *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, New York.
- Lawler E.L. 1978. "Sequencing Jobs to Minimize Total Weighted Completion Time subject to Precedence Constraints", *Annals of Discrete Mathematics* 2, 75–90.
- Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G., Shmoys D.B. 1993. "Sequencing and Scheduling: Algorithms and Complexity", in S.C. Graves, A.H.G. Rinnooy Kan, P.H. Zipkin, eds., *Logistics of Production and Inventory, Handbooks in Operations Research and Management*

Science 4, North-Holland, Amsterdam, The Netherlands, 445–522.

McCormick, S.T. 1998. “Fast Algorithms for Parametric Scheduling Come from Extensions to Parametric Maximum Flow”, *Operations Research* 47, 744-756.

Phillips C., Stein C., Wein J. 1998. “Minimizing Average Completion Time in the Presence of Release Dates”, *Mathematical Programming* 82, 199–223.

Picard J.-C. 1976. “Maximum closure of a graph and application to combinatorial problems,” *Management Science* 22, 1268–1272.

Picard J.-C., Queyranne M. 1980. “On the Structure of All Minimum Cuts in a Network and Applications”, *Mathematical Programming Study* 13, 8–16.

Potts C.N. 1991. “A Lagrangean Based Branch-and-Bound Algorithm for Single Machine Sequencing with Precedence Constraints to Minimize Total Weighted Completion Time”, *Management Science* 31, 1300-1311.

Queyranne M. 1993. “Structure of a Simple Scheduling Polyhedron”, *Mathematical Programming* 58 263–285.

Queyranne M., Schulz A.S. 1996. “Polyhedral Approaches to Machine Scheduling”, Report 408/1994, Department of Mathematics, University of Technology, Berlin, Germany, November 1994; revised, October 1996.

Queyranne M., Wang Y. 1991a. “Single-Machine Scheduling Polyhedra with Precedence Constraints”, *Mathematics of Operations Research* 16, 1–20.

Queyranne M., Wang Y. 1991b. “A Cutting Plane Procedure for Precedence-Constrained Single Machine Scheduling”, Working paper, Faculty of Commerce, University of British Columbia, Vancouver, Canada, August 1991.

Schulz A.S. 1996. “Scheduling to Minimize Total Weighted Completion Time: Performance

Guarantees of LP-Based Heuristics and Lower Bounds”, in W.H. Cunningham, S.T. McCormick, M. Queyranne, eds., *Integer Programming and Combinatorial Optimization*, LNCS 1084, Springer, Berlin, Germany, 301–315.

Sidney J.B. 1975. “Decomposition Algorithms for Single-Machine Sequencing with Precedence Relations and Deferral Costs”, *Operations Research* 23, 283–298.

Wolsey L.A. 1990. “Formulating Single Machine Scheduling Problems With Precedence Constraints”, *Economic Decision-Making: Games, Econometrics and Optimisation*, J.J. Gabszewicz, J.-F. Richard, L.A. Wolsey (Eds.), North-Holland, Amsterdam, 473–484.

n	inst.	a_gap	m_gap	a_gapim	a_gap1	a_r_wp	a_mpc	m_mpc	a_cpu	m_cpu
100	1- 2	0.00	0.00	0.00	0.00	100.00	3.00	3	0.28	0.28
	3-10	0.74	2.27	72.86	31.38	98.02	29.00	59	0.26	0.33
	11-18	2.16	3.51	47.97	13.82	94.82	55.12	79	0. 17	0.20
	19-20	0.36	0.63	33.53	35.86	98.48	22.50	28	0.17	0.20
500	1- 2	0.00	0.00	50.00	50.00	100.00	3.50	4	1.60	1.63
	3-10	1.48	2.35	50.77	5.57	97.87	248.50	408	1.53	1.83
	11-18	1.20	2.46	32.97	8.64	97.76	323.00	460	1. 13	1.48
	19-20	0.09	0.15	41.23	24.48	99.64	53.00	76	1.00	1.03
1000	1- 2	0.00	0.00	74.01	43.24	99.99	6.50	7	3.67	3.70
	3-10	1.31	2.44	41.12	3.78	98.28	618.62	881	4.87	6.83
	11-18	1.42	3.48	20.48	5.51	97.80	579.12	950	3 .32	4.80
	19-20	0.03	0.04	46.26	28.83	99.86	38.50	55	2.40	2.42
1500	1- 2	0.00	0.00	83.05	64.73	99.98	13.00	15	6.36	6.37
	3-10	1.00	1.33	40.20	3.36	98.72	991.88	1422	9.66	15.95
	11-18	0.77	1.49	20.50	5.15	98.74	802.25	1091	5 .18	6.35
	19-20	0.02	0.03	44.27	24.34	99.88	48.00	57	4.02	4.07
2000	1- 2	0.00	0.00	78.41	45.86	99.98	13.50	16	10.40	10.68
	3-10	0.93	1.30	36.73	2.76	98.84	1027.12	1381	14.05	26.93
	11-18	1.13	1.92	13.23	3.03	98.27	1280.00	1933	9.79	13.37
	19-20	0.01	0.02	49.21	28.40	99.93	54.50	63	6.76	6.85

Table 1: Aggregated results