Single machine precedence constrained scheduling is a vertex cover problem

Christoph Ambühl
University of Liverpool
Liverpool, United Kingdom
christoph@csc.liv.ac.uk

Monaldo Mastrolilli
IDSIA
Manno-Lugano, Switzerland
monaldo@idsia.ch

September 28, 2007

Abstract

In this paper we study the single machine precedence constrained scheduling problem of minimizing the sum of weighted completion time. Specifically, we settle an open problem first raised by Chudak & Hochbaum and whose answer was subsequently conjectured by Correa & Schulz. As shown by Correa & Schulz, the proof of this conjecture implies that the addressed scheduling problem is a special case of the vertex cover problem. This means that previous results for the scheduling problem can be explained, and in some cases improved, by means of vertex cover theory. For example, the conjecture implies the existence of a polynomial time algorithm for the special case of two-dimensional partial orders. This considerably extends Lawler's result from 1978 for series-parallel orders.

1 Introduction

We address the problem of scheduling a set $N = \{1, \ldots, n\}$ of n jobs on a single machine. The machine can process at most one job at a time. Each job j is specified by its length p_j and its weight w_j, where p_j and w_j are nonnegative integers. We only consider *non-preemptive* schedules, in which all p_j units of job j must be scheduled consecutively. Jobs have precedence constraints between them that are specified in the form of a directed acyclic graph $G = (N, P)$, where $(i, j) \in P$ implies that job i must be completed before job j can be started. We assume that G is transitively closed, i.e., if $(i, j), (j, k) \in P$ then $(i, k) \in P$. The goal is to find a schedule which minimizes the sum $\sum_{j=1}^{n} w_j C_j$, where C_j is the time at which job j completes in the given schedule. In standard scheduling notation (see e.g. Graham et al. [10]), this problem is known as $1|prec\,|\sum w_j C_j$. The general version of $1|prec\,|\sum w_j C_j$ was shown to be strongly NP-hard by Lawler [15] and Lenstra & Rinnooy Kan [17].

Nevertheless, special cases are known to be polynomial-time solvable. In 1956, Smith [28] showed that, in absence of precedence constraints, an optimal solution could be found by sequencing the jobs in non-increasing order of the ratio w_i/p_i. Afterwards, several other results for special classes of precedence constraints were proposed. The most important class of instances known to be polynomially solvable to date are those with series-parallel precedence constraints. For this class, Lawler [15] gave an $O(n \log n)$ time algorithm. Goemans & Williamson [9] provided a nice alternative proof for the correctness of Lawler's algorithm by using a two-dimensional Gantt chart. Several authors worked on finding larger classes of polynomially solvable instances, mainly by considering precedence constraints which are lexicographic sums [30] of polynomially solvable classes (see [16] for a survey).

Two-dimensional partial orders represent an important generalization of the series-parallel case [19]. However, determining its complexity is a long-standing open problem. A first attempt [29] dates back to 1984, and the currently best known approximation factor is 3/2 [7]. Other restricted classes of precedence constraints, such as interval orders and convex bipartite precedence constraints, have been studied (see e.g. [19] for a survey, and [7, 14, 31] for more recent results). Woeginger [31] proved that the general case of $1|prec|\sum w_j C_j$ is not harder to approximate than some fairly restricted special cases, among them for example the case of bipartite precedence constraints where all jobs on the first partition class have processing time 1 and weight 0, and all jobs on the second partition class have weight 1 and processing time 0.

For the general version of $1|prec|\sum w_j C_j$, closing the approximability gap is considered an outstanding open problem in scheduling theory (see e.g. [26]). Only very recently it was proved that the problem does not admit a PTAS, unless NP-complete problems can be solved in randomized subexponential time [3]. On the positive side, several polynomial time 2-approximation algorithms are known. Pisaruk [23] claims to have obtained the first of such 2-approximation algorithms. Schulz [25] and Hall, Schulz, Shmoys & Wein [11] gave 2-approximation algorithms by using a linear programming relaxation in completion time variables. Chudak & Hochbaum [6] gave another algorithm based on a relaxation of the linear program studied by Potts [24]. Independently, Chekuri & Motwani [5] and Margot, Queyranne & Wang [18], provided identical, extremely simple 2-approximation algorithms based on Sidney's decomposition theorem [27] from 1975.

A *Sidney decomposition* partitions the set N of jobs into sets $S_1, S_2, \dots, S_k$ by using a generalization of Smith's rule [28], such that there exists an optimal schedule where jobs from S_i are processed before jobs from S_{i+1}, for any $i = 1, \dots, k-1$. Lawler [15] showed that a Sidney decomposition can be computed in polynomial time by performing a sequence of min-cut computations. Chekuri & Motwani [5] and Margot, Queyranne & Wang [18] actually proved that every schedule that complies with a Sidney decomposition is a 2-approximate solution. Correa & Schulz [7] subsequently showed that all known 2-approximation algorithms follow a Sidney decomposition, and therefore belong to the class of algorithms described by Chekuri & Motwani [5] and Margot, Queyranne & Wang [18].

This result is rather demoralizing, as it shows that despite of many years of active research, all the approximation algorithms rely on the Sidney decomposition dating back to 1975. It should be emphasized that the Sidney decomposition does not impose any ordering among the jobs within a set S_i, and any ordering will do just fine for a 2-approximation. This shows that we basically have no clue how to order the jobs within the sets S_i.

The current state of $1|prec|\sum w_j C_j$, in terms of approximation, is very similar to one of the most famous and best studied NP-hard problems: the vertex cover problem (see [22] for a survey). Despite considerable efforts, the best known approximation algorithm still has a ratio of $2 - o(1)$. Improving this ratio is generally considered one of the most outstanding open problems in theoretical computer science. Hochbaum [13] conjectured that it is not possible to obtain a better factor.

In 1973, Nemhauser & Trotter [20, 21] used the following integer program to model the minimum vertex cover problem in a weighted graph (V, E) with weights w_i on the vertices.

$$\begin{array}{lll} \text{[VC-IP]} \qquad & \min & \displaystyle\sum_{i \in V} w_i x_i & \\ & \text{s.t.} & x_i + x_j \geq 1 & \{i, j\} \in E \\ & & x_i \in \{0, 1\} & i \in V \end{array}$$

They also studied the linear relaxation [VC-LP] of [VC-IP], and proved that any basic feasible solution for [VC-LP] is *half-integral*, that is $x_i \in \{0, \frac{1}{2}, 1\}$ for all $i \in V$. Moreover, they showed [21] that those variables which assume binary values in an optimal solution for [VC-LP] retain the same value in an optimal solution for [VC-IP]. This is known as the *persistency property* of vertex cover, and a solution is said to *comply* with the persistency property if it retains the binary values of an optimal solution for [VC-LP]. Hochbaum [13] pointed out that any feasible solution that complies with the persistency property is a 2 approximate solution.

2 Results and Implications

The dominant role the Sidney decomposition plays in $1|prec|\sum w_j C_j$ seems to be very well reflected by the persistency property for the vertex cover problem. It was often suspected that there is a strong relationship between vertex cover and $1|prec|\sum w_j C_j$ [26].

In this paper we show that this speculation is justified. We hope that this result will be an important step towards a proof that both problems are equivalent in terms of approximability. Proving this would give a more or less satisfactory answer to the ninth problem of the famous ten open problems in scheduling theory [26]. In general terms, building on previous work by Potts [24], Chudak & Hochbaum [6] and Correa & Schulz [7], we provide the missing link to the proof of the following result.

Theorem 1. *$1|prec|\sum w_j C_j$ is a special case of the vertex cover problem.*

Given that many people have worked hard on improving the various 2-approximation algorithms of $1|prec|\sum w_j C_j$, it seems likely that $1|prec|\sum w_j C_j$ is as hard to approximate as vertex cover. Certainly people working on approximation algorithms for vertex cover should try their luck at $1|prec|\sum w_j C_j$ first. In this vein and by using the results of this paper, Svensson and the authors have recently proved that the considered scheduling problem is as hard to approximate as Vertex Cover when the so-called fixed cost is disregarded from the objective function [3]. The fixed cost of a schedule is the part of the objective function which only depends on the problem instance, but not on the schedule.

Theorem 1 is proved by showing that an optimal solution for Potts' integer program ([P-IP]) is also optimal for the integer program of Chudak & Hochbaum ([CH-IP]). This solves an open problem posted in [6] whose answer was conjectured in [7]. Pott's IP is known to model $1|prec\,|\sum w_jC_j$ correctly, whereas [CH-IP] is a relaxation of [P-IP] obtained by removing a big chunk of the conditions from [P-IP]. Theorem 1 then follows from the equivalence of [CH-IP] and the integer program [CS-IP] of Correa & Schulz [7], which is a special case of [VC-IP]. We prove the same result also for the linear relaxations of the three IPs, which are subsequently denoted by [P-LP], [CH-LP], and [CS-LP].

Apart from the long term consequences of our result, there are a few direct ones. Since $1|prec\,|\sum w_jC_j$ is a special case of the vertex cover problem, any approximation algorithm for vertex cover translates into an approximation algorithm for $1|prec|\sum w_jC_j$ of the same approximation guarantee.

More importantly, our result considerably increases the class of instances that can be solved optimally in polynomial time to the class of two-dimensional partial orders. A partial order (N, P) has dimension two if P can be described as the intersection of two total orders of N (see [19] for a survey).

In [7] it is proved that the vertex cover graph associated with [CS-LP] is bipartite if and only if the precedence constraints are of dimension at most two. This means that every basic feasible solution to [CS-LP] is integral in this case. They also give a 3/2-approximation algorithm for the two-dimensional case, which improves on a previous result [14].

In this paper we show that any integral solution to [CH-LP] can be converted into a feasible solution for [P-IP] without deteriorating the objective function value. Together with a result of [7], this implies that instances with two-dimensional precedence constraints are solvable in polynomial time. We emphasize that series-parallel partial orders have dimension at most two, but the class of two-dimensional partial orders is substantially larger [4]. Thus, the polynomial-time solvability of the two-dimensional case considerably extends Lawler's result [15] from 1978 for series-parallel orders.

The result described in this paper has triggered new results exploring a connection between $1|prec\,|\sum w_jC_j$ and dimension theory of partial orders [2, 1]. This has led to improved and new approximation algorithms for many special cases of $1|prec\,|\sum w_jC_j$.

Indeed, it turns out the graph of the [CS-IP] is a very well studied structure in dimension theory of partial orders: the graph of incomparable pairs [30]. Figure 1 makes use of this insight to illustrate Theorem 1 with a small example. The $1|prec\,|\sum w_jC_j$ instance on the left has four jobs, each of them has a processing time p_i and a weight w_i, which are denoted to the right of each job. The graph on the right is the so-called graph of incomparable pairs G_P. It has a vertex for every pair of jobs (i, j) not connected by a precedence constraint, and an edge between vertices (i_1, j_1) and (i_2, j_2) if adding the precedence constraints (j_1, i_1) and (j_2, i_2) would create a cycle in the precedence graph. The weight of a vertex (i, j) is $p_i \cdot w_j$ (denoted on the right of each vertex). The minimum (weighted) vertex cover of G_P corresponds to an optimal solution of [CS-IP], which can then be turned into an optimal schedule by Theorem 1. In the example, the minimum weighted vertex cover is represented by the gray vertices. It corresponds to the schedule 3,4,1,2 for the $1|prec\,|\sum w_jC_j$ instance.

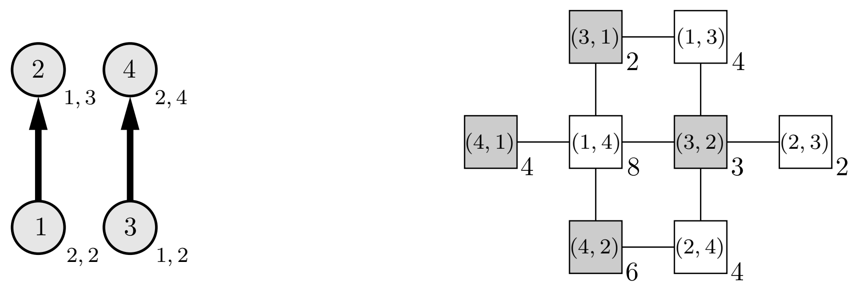

Figure 1: An Example illustrating Theorem 1.

3 Preliminaries

To simplify notation, we implicitly assume hereafter that tuples and sets of jobs have no multiplicity. Therefore, $(a_1, a_2, \dots, a_k) \in N^k$ and $\{b_1, b_2, \dots, b_k\} \subseteq N$ denote a tuple and a set, respectively, with k distinct elements.

In the following, we introduce several linear programming formulations and relaxations of $1|prec|\sum w_j C_j$ using linear ordering variables δ_{ij}. The variable δ_{ij} has value 1 if job i precedes job j in the corresponding schedule, and 0 otherwise. The first formulation using linear ordering variables is due to Potts [24], and it can be stated as follows.

$$
\begin{aligned}
\text{[P-IP]} \qquad & \min \quad \sum_{j \in N} p_j w_j + \sum_{(i,j) \in N^2} \delta_{ij} p_i w_j && && (1)\\
& \text{s.t.} \quad \delta_{ij} + \delta_{ji} = 1 && \{i,j\} \subseteq N && (2)\\
& \qquad \delta_{ij} = 1 && (i,j) \in P && (3)\\
& \qquad \delta_{ij} + \delta_{jk} + \delta_{ki} \leq 2 && (i,j,k) \in N^3 && (4)\\
& \qquad \delta_{ij} \in \{0,1\} && (i,j) \in N^2 && (5)
\end{aligned}
$$

Constraint (2) ensures that either job i is scheduled before j or viceversa. If job i is constrained to precede j in the partial order P, then this is seized by Constraint (3). The set of Constraints (4) is used to capture the transitivity of the ordering relations (i.e., if i is scheduled before j and j before k, then i is scheduled before k). It is easy to see that [P-IP] is indeed a complete formulation of the problem [24].

Chudak & Hochbaum [6] suggested to study the following relaxation of [P-IP]:

$$
\begin{aligned}
\text{[CH-IP]} \qquad \min\ (1) \quad \text{s.t.} \quad & (2), (3), (5) && \\
& \delta_{jk} + \delta_{ki} \leq 1 \qquad (i,j) \in P, \{i,j,k\} \subseteq N && (6)
\end{aligned}
$$

In [CH-IP], Constraints (4) are replaced by Constraints (6). These inequalities correspond in general to a proper subset of (4), since only those transitivity Constraints (4) for which two of the participating jobs are already related to each other by a precedence constraint are kept. If the Integrality Constraints (5) are relaxed and replaced by

$$\delta_{ij} \geq 0 \qquad (i,j) \in N^2, \qquad (7)$$

then we will refer to the linear relaxations of [P-IP] and [CH-IP] as [P-LP] and [CH-LP], respectively. Chudak & Hochbaum's formulations lead to two natural open questions, first raised by Chudak & Hochbaum [6] and whose answers were conjectured by Correa & Schulz [7]:

Conjecture 1. *[7] An optimal solution to [P-IP] is also optimal for [CH-IP].*

Conjecture 2. *[7] An optimal solution to [P-LP] is also optimal for [CH-LP].*

The correctness of these conjectures has several important consequences, the most prominent being that the addressed problem can be seen as a special case of the vertex cover problem. Indeed, Correa & Schulz proposed the following linear ordering relaxation of [P-IP] that can be interpreted as a vertex cover problem [7]:

$$
\begin{array}{lllll}
\text{[CS-IP]} & \min & \displaystyle\sum_{j\in N} p_j w_j + \sum_{(i,j)\in N^2} \delta_{ij} p_i w_j & & \\
& \text{s.t.} & \delta_{ij} + \delta_{ji} \geq 1 & & \{i,j\} \subseteq N \\
& & \delta_{ik} + \delta_{kj} \geq 1 & & (i,j) \in P, \{i,j,k\} \subseteq N \\
& & \delta_{i\ell} + \delta_{kj} \geq 1 & & (i,j),(k,\ell) \in P, \{i,j,k,\ell\} \subseteq N \\
& & \delta_{ij} = 1, \delta_{ji} = 0 & & (i,j) \in P \\
& & \delta_{ij} \in \{0,1\} & & (i,j) \in N^2
\end{array}
$$

As usual, let us denote by [CS-LP] the linear relaxation of [CS-IP]. The following result is implied in [7].

Theorem 2. *[7] The optimal solutions to [CH-LP] and [CS-LP] coincide. Moreover, any feasible solution to [CS-LP] can be transformed in $O(n^2)$ time into a feasible solution to [CH-LP] without increasing the objective value. Both statements are true for [CH-IP] and [CS-IP] as well.*

As [CS-IP] represents an instance of the vertex cover problem, it follows from the work of Nemhauser & Trotter [20, 21] that [CS-LP] is half-integral, and that an optimal solution can be obtained via a single min-cut computation. Hence, the same holds for [CH-LP].

Theorem 3. *[6, 7] The linear programs [CH-LP] and [CS-LP] are half-integral.*

In order to unify the IP and the LP versions of the conjectures, the following parameterized setting is considered throughout this paper. We introduce yet another version of [P] and [CH]. For a parameter $\Delta \in \{\frac{1}{2}, 1\}$, let [CH-$\Delta$] and [P-$\Delta$] be equal to [CH-LP] and [P-LP], respectively, but with the additional constraint that all δ_{ij} are multiples of Δ. For Conjecture 1, it is obvious that [CH-IP] is equivalent to [CH-1]. The same holds for [P-IP] and [P-1]. As far as Conjecture 2 is concerned, Theorem 3 implies that any optimal solution for [CH-$\frac{1}{2}$] is optimal for [CH-LP] as well. Also, any feasible solution for [P-$\frac{1}{2}$] is feasible for [P-LP].

4 Main Theorem and Proof Overview

Our main theorem can be stated as follows.

Theorem 4. *Any feasible solution for [CH-Δ] can be turned into a feasible solution for [P-Δ] in $O(n^3)$ time without increasing the objective value.*

Theorem 4 represents the missing link between problem $1|prec|\sum w_j C_j$ and vertex cover. With Theorem 4 in place, the claim of Theorem 1 directly follows by using Theorem 2. Moreover, both Conjecture 1 and 2 are proved to be true as corollaries. The proof of Theorem 4 is given in the following.

Proof overview. Let vector $\delta = \big(\delta_{ij} : (i,j) \in N^2\big)$ denote a feasible solution to [CH-Δ] throughout the paper. For any $(i,j,k) \in N^3$, let $\langle ijk \rangle$ denote the set $\{(i,j),(j,k),(k,i)\}$. Let us call $\langle ijk \rangle$ an *oriented 3-cycle*. Moreover, let $\mathcal{C}$ be the set of all oriented 3-cycles of the set N. Note that $\langle ijk \rangle = \langle jki \rangle = \langle kij \rangle$ and $\langle jik \rangle = \langle kji \rangle = \langle ikj \rangle$, but $\langle ijk \rangle \neq \langle jik \rangle$. In contrast to $\langle ijk \rangle \in \mathcal{C}$, any reordering of the jobs in $(i,j,k) \in N^3$ results in a different triple. The following values are used to measure the infeasibility of δ for [P-Δ].

$$\alpha_{\langle ijk \rangle} := \max\left(0, \delta_{ij} + \delta_{jk} + \delta_{ki} - 2\right) \qquad \text{for all } \langle ijk \rangle \in \mathcal{C} \tag{8}$$

Observe that δ is feasible for [P-Δ] if and only if $\alpha_{\langle ijk \rangle} = 0$ for all $\langle ijk \rangle \in \mathcal{C}$, otherwise the transitivity Constraint (4) is violated. Let α be the total sum of all $\alpha_{\langle ijk \rangle}$ values.

$$\alpha := \sum_{\langle ijk \rangle \in \mathcal{C}} \alpha_{\langle ijk \rangle} \tag{9}$$

Let us call α the *total infeasibility* of solution δ.

For any job k and $1 \leq s \leq 1/\Delta$, we also define the following set of job pairs that together with k violate Constraint (4).

$$\mathcal{B}_s^{(k)} := \left\{(i,j) : \alpha_{\langle ijk \rangle} \geq s\Delta\right\} \qquad \text{for all } k \in N \text{ and } 1 \leq s \leq 1/\Delta \tag{10}$$

These sets will be used to alter a feasible solution δ towards [P-Δ] feasibility, as explained in the following. Note that δ is feasible for [P-Δ] if and only if all $\mathcal{B}_s^{(k)}$ are empty.

Let $\mathfrak{B} := \left\{\mathcal{B}_s^{(k)} : k \in N \text{ and } 1 \leq s \leq 1/\Delta\right\}$. For any set $\mathcal{B} \in \mathfrak{B}$, consider the vector $\delta^{\mathcal{B}}$ defined as:

$$\delta_{ij}^{\mathcal{B}} := \begin{cases} \delta_{ij} - \Delta & \text{if } (i,j) \in \mathcal{B}; \\ \delta_{ij} + \Delta & \text{if } (j,i) \in \mathcal{B}; \\ \delta_{ij} & \text{otherwise}; \end{cases} \qquad \text{for all } (i,j) \in N^2. \tag{11}$$

We will show later in Corollary 7 that $(i,j) \in \mathcal{B}$ and $(j,i) \in \mathcal{B}$ cannot hold at the same time, therefore $\delta^{\mathcal{B}}$ is well defined.

For any $\langle ijk \rangle \in \mathcal{C}$, let $\alpha_{\langle ijk \rangle}^{\mathcal{B}}$ and $\alpha^{\mathcal{B}}$ be the values defined in (8) and in (9), respectively, for $\delta^{\mathcal{B}}$.

The following lemma plays a fundamental role in the proof of Theorem 4. It asserts that when δ is not feasible for [P-Δ], it is always possible to choose a set $\mathcal{B} \in \mathfrak{B}$ such that $\delta^{\mathcal{B}}$ has a lower total infeasibility and a not larger objective value. Its proof is given in the remainder part of the paper.

Lemma 5. *Let δ be a feasible solution to [CH-Δ], which is not feasible for [P-Δ]. Then*

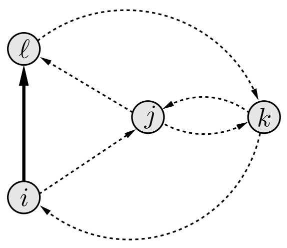

Figure 2: Illustration for Lemma 6

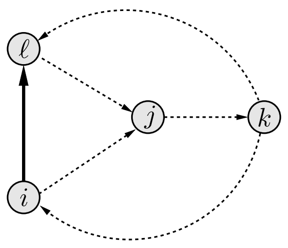

Figure 3: Illustration for Lemma 8

(a) Vector $\delta^{\mathcal{B}}$ is a feasible solution to [CH-Δ] for all $\mathcal{B} \in \mathfrak{B}$.

(b) There exists a nonempty set $\mathcal{B} \in \mathfrak{B}$ such that the objective value of $\delta^{\mathcal{B}}$ is not larger than the objective value of δ.

(c) For any $\mathcal{B} \in \mathfrak{B}$, we have $\alpha^{\mathcal{B}} \leq \alpha - |\mathcal{B}|\Delta$.

By using Lemma 5, the proof of Theorem 4 becomes straightforward.

Proof of Theorem 4. The parts (a) and (b) of Lemma 5 ensure that among all the solutions $\delta^{\mathcal{B}}$, with $\mathcal{B} \in \mathfrak{B}$ and $\mathcal{B} \neq \emptyset$, there is one whose objective value is not larger than δ.

What is more, part (c) even ensures that the total infeasibility value α decreases by $|\mathcal{B}|\Delta$ when moving from δ to $\delta^{\mathcal{B}}$. Since $|\mathcal{B}|\Delta \geq \Delta$ and $\alpha_{\langle ijk \rangle} \geq 0$, for any $\langle ijk \rangle \in \mathcal{C}$, repeating this transformation will eventually lead to a solution for which (9) evaluates to zero, which means that it is feasible for [P-Δ]. An algorithm which performs this transformation in time $O(n^3)$ is described in Section 7. ☐

5 Two Useful Lemmas

In the following, we provide two properties of the α-values that will prove useful in the proof of Lemma 5.

Lemma 6. *For any $\langle ijk \rangle, \langle j\ell k \rangle \in \mathcal{C}$, if $(i, \ell) \in P$ or $i = \ell$ then*

$$\min\left(\alpha_{\langle ijk \rangle}, \alpha_{\langle j\ell k \rangle}\right) = 0.$$

Proof. See Figure 2. The proof is by contradiction. Assume $\alpha_{\langle ijk \rangle} > 0$ and $\alpha_{\langle j\ell k \rangle} > 0$. Then

$$\delta_{ij} + \delta_{jk} + \delta_{ki} > 2 \quad \text{and} \quad \delta_{j\ell} + \delta_{\ell k} + \delta_{kj} > 2.$$

By adding up the previous two inequalities, and using Constraint (2), we obtain

$$\delta_{ij} + \delta_{j\ell} + \delta_{\ell k} + \delta_{ki} > 3,$$

which is impossible since by Constraints (2), (6) and (7), we have $\delta_{ij} + \delta_{j\ell} \leq 2$ and $\delta_{\ell k} + \delta_{ki} \leq 1$. ☐

The following corollary follows easily from Lemma 6 and the definition of $\mathfrak{B}$.

Corollary 7. *For any $\{i,j\} \subseteq N$ and $\mathcal{B} \in \mathfrak{B}$, at most one pair between (i,j) and (j,i) belongs to set $\mathcal{B}$.*

Lemma 8. *For any $\langle ijk\rangle, \langle \ell jk\rangle \in \mathcal{C}$ with $(i,\ell) \in P$, if $\delta_{ij} = \delta_{\ell j}$ (or equivalently $\delta_{ji} = \delta_{j\ell}$ by Constraint (2)) then $\alpha_{\langle ijk\rangle} \leq \alpha_{\langle \ell jk\rangle}$ and $\alpha_{\langle jik\rangle} \geq \alpha_{\langle j\ell k\rangle}$.*

Proof. See Figure 3. By Constraints (2) and (6) we have $\delta_{k\ell} \geq \delta_{ki}$ and $\delta_{ik} \geq \delta_{\ell k}$, that, by the assumptions (i.e., $\delta_{ij} = \delta_{\ell j}$ and $\delta_{ji} = \delta_{j\ell}$), imply

$$\begin{aligned} \delta_{ij} + \delta_{jk} + \delta_{ki} &\leq \delta_{\ell j} + \delta_{jk} + \delta_{k\ell}, \\ \delta_{ji} + \delta_{kj} + \delta_{ik} &\geq \delta_{j\ell} + \delta_{kj} + \delta_{\ell k}. \end{aligned}$$

The claim follows by (8). □

6 Proof of Lemma 5

Proof of Lemma 5(a). To prove that claim, we fix an arbitrary set $\mathcal{B}_s^{(k)} \in \mathfrak{B}$. To ease notation, we will often write $\mathcal{B}$ instead of $\mathcal{B}_s^{(k)}$. The claim follows by showing that the solution $\delta^{\mathcal{B}}$ satisfies Constraints (2), (3), (6) and (7), and that all $\delta_{ij}^{\mathcal{B}}$ are multiples of Δ. The latter and Constraints (2) directly follow from the feasibility of δ and Transformation (11).

Constraints (3) hold since for $(i,j) \in P$, we have $\alpha_{\langle ijk\rangle} = 0$ and $\alpha_{\langle jik\rangle} = 0$ and therefore neither (i,j) nor (j,i) will be part of the set $\mathcal{B}_s^{(k)}$, which ensures $\delta_{ij}^{\mathcal{B}} = \delta_{ij} = 1$.

Regarding Constraints (6)

$$\delta_{\ell j} + \delta_{ji} \leq 1 \qquad (i,\ell) \in P, \{i,j,\ell\} \subseteq N,$$

we distinguish the two complementary cases. In the first case, we assume $\delta_{\ell j} + \delta_{ji} = 1$. We then obviously have $\delta_{ij} = \delta_{\ell j}$ and $\delta_{ji} = \delta_{j\ell}$ by Constraints (2). By definition of $\delta^{\mathcal{B}}$, in order to violate the constraint, we need at least one of (i,j) and (j,ℓ) to be in $\mathcal{B}_s^{(k)}$. If $(i,j) \in \mathcal{B}_s^{(k)}$ (and therefore $(j,i) \notin \mathcal{B}_s^{(k)}$ by Corollary 7), then we have $\alpha_{\langle ijk\rangle} \leq \alpha_{\langle \ell jk\rangle}$ by Lemma 8, that implies $(\ell,j) \in \mathcal{B}_s^{(k)}$ (and $(j,\ell) \notin \mathcal{B}_s^{(k)}$). Similarly, if $(j,\ell) \in \mathcal{B}_s^{(k)}$ (and therefore $(\ell,j) \notin \mathcal{B}_s^{(k)}$), then by Lemma 8 we have $\alpha_{\langle jik\rangle} \geq \alpha_{\langle j\ell k\rangle}$, that implies $(j,i) \in \mathcal{B}_s^{(k)}$ (and $(i,j) \notin \mathcal{B}_s^{(k)}$). Hence in the case $\delta_{\ell j} + \delta_{ji} = 1$, Constraints (6) are satisfied since we have $\delta_{ij}^{\mathcal{B}} + \delta_{j\ell}^{\mathcal{B}} \leq \delta_{ij} + \delta_{j\ell}$.

In the second case we assume $\delta_{\ell j} + \delta_{ji} < 1$ and argue as follows. If $(i,j) \in \mathcal{B}_s^{(k)}$ then $\alpha_{\langle ijk\rangle} > 0$, and by Lemma 6 it is $\alpha_{\langle j\ell k\rangle} = 0$, which implies $(j,\ell) \notin \mathcal{B}_s^{(k)}$. By these arguments, it is easy to see that $\delta_{\ell j}^{\mathcal{B}} + \delta_{ji}^{\mathcal{B}} \leq \delta_{\ell j} + \delta_{ji} + \Delta \leq 1$. This completes the proof for Constraints (6).

Finally, consider the nonnegativity Constraint (7). Note that $(i,j) \in \mathcal{B}_s^{(k)}$ implies $\alpha_{\langle ijk\rangle} \geq \Delta$, by the feasibility of δ. This in turn means $\delta_{ij} \geq \Delta$. The latter ensures that any $\delta_{ij}^{\mathcal{B}}$ satisfies Constraint (7). □

Lemma 9.

$$\sum_{(i,j,k)\in N^3} \alpha_{\langle ijk\rangle} \cdot p_k p_i w_j = \sum_{(i,j,k)\in N^3} \alpha_{\langle ijk\rangle} \cdot p_k p_j w_i$$

Proof. By Definition (8) the following identities hold.

$$\begin{aligned}
\sum_{(i,j,k)\in N^3} \alpha_{\langle ijk\rangle} \cdot p_k p_i w_j &= \sum_{\langle ijk\rangle\in\mathcal{C}} \alpha_{\langle ijk\rangle} \left(p_k p_i w_j + p_j p_k w_i + p_i p_j w_k\right) \\
&= \sum_{\langle ijk\rangle\in\mathcal{C}} \alpha_{\langle ijk\rangle} \left(p_i p_k w_j + p_k p_j w_i + p_j p_i w_k\right) \\
&= \sum_{(i,j,k)\in N^3} \alpha_{\langle ijk\rangle} \cdot p_k p_j w_i
\end{aligned}$$

□

Proof of Lemma 5(b). We first observe that if we decrease the value of any δ_{ij} by Δ and increase δ_{ji} by Δ, then the difference between the new objective value and the previous one is equal to $\Delta \cdot (p_j w_i - p_i w_j)$. Now, for any $\mathcal{B} \in \mathfrak{B}$, let $V(\mathcal{B})$ be defined as follows.

$$V(\mathcal{B}) := \sum_{(i,j)\in\mathcal{B}} p_j w_i - \sum_{(i,j)\in\mathcal{B}} p_i w_j \tag{12}$$

If we have a look at Transformation (11), the difference between the objective values of $\delta^{\mathcal{B}}$ and δ, respectively, is equal to $\Delta \cdot V(\mathcal{B})$.

If solution δ is not feasible for [P-Δ], what we have to show is that there always exists a set $\mathcal{B} \neq \emptyset$ (with $\mathcal{B} \in \mathfrak{B}$) for which (12) is non-positive. We prove this by contradiction.

Let us first define the following indicator function.

$$\mathcal{T}(S) := \begin{cases} 1 & \text{if the statement } S \text{ is true;} \\ 0 & \text{otherwise.} \end{cases}$$

Recall that by the definition of $\mathcal{B}_s^{(k)}$ in (10), it holds $(i,j) \in \mathcal{B}_s^{(k)}$ if and only if $\alpha_{\langle ijk\rangle} \geq s\Delta$. This allows to write

$$\alpha_{\langle ijk\rangle} = \Delta \cdot \sum_{s=1}^{1/\Delta} \mathcal{T}\left((i,j) \in \mathcal{B}_s^{(k)}\right).$$

Now, assume that for all $\mathcal{B} \neq \emptyset$ the value $V(\mathcal{B})$ is positive, i.e., the following inequality holds.

$$\sum_{(i,j)\in\mathcal{B}} p_i w_j < \sum_{(i,j)\in\mathcal{B}} p_j w_i \qquad \text{for all nonempty } \mathcal{B} \in \mathfrak{B} \tag{13}$$

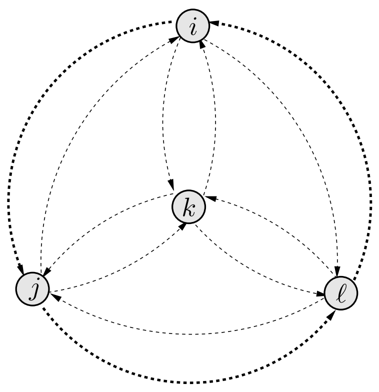

Figure 4: Illustration for case (1) of the proof of Lemma 5(c)

Using Assumption (13), we can conclude

$$\begin{aligned}
\sum_{(i,j,k)\in N^3} \alpha_{\langle ijk\rangle} \cdot p_k p_i w_j &= \sum_{(i,j,k)\in N^3} p_k \left(\Delta \cdot \sum_{s=1}^{1/\Delta} \mathcal{T}\left((i,j) \in \mathcal{B}_s^{(k)} \right) \right) p_i w_j \\
&= \sum_{k\in N} p_k \Delta \sum_{s=1}^{1/\Delta} \left(\sum_{(i,j)\in \mathcal{B}_s^{(k)}} p_i w_j \right) \\
&< \sum_{k\in N} p_k \Delta \sum_{s=1}^{1/\Delta} \left(\sum_{(i,j)\in \mathcal{B}_s^{(k)}} p_j w_i \right) \qquad (14) \\
&= \sum_{(i,j,k)\in N^3} p_k \left(\Delta \cdot \sum_{s=1}^{1/\Delta} \mathcal{T}\left((i,j) \in \mathcal{B}_s^{(k)} \right) \right) p_j w_i \\
&= \sum_{(i,j,k)\in N^3} \alpha_{\langle ijk\rangle} \cdot p_k p_j w_i,
\end{aligned}$$

which clearly contradicts Lemma 9. The interesting line here is the strict Inequality (14). If we assume $p_x > 0$ for all jobs $x \in N$, it just follows from Assumption (13) and the obvious fact that $V(\mathcal{B}) = 0$ when $\mathcal{B}$ is an empty set.

To prove it also for the case with zero processing times, we need to show that there exists a job $x \in N$ and $s \in \{1, \dots, 1/\Delta\}$ with $p_x > 0$ and $\mathcal{B}_s^{(x)} \neq \emptyset$. With this aim, consider any nonempty set $\mathcal{B}_1^{(k)}$. Note that the assumption that δ is not feasible for [P-Δ] guarantees the existence of set $\mathcal{B}_1^{(k)} \neq \emptyset$. Because of Assumption (13), there must be at least one $(i,j) \in \mathcal{B}_1^{(k)}$ with $p_i w_j < p_j w_i$, which implies $p_j > 0$. It also follows from $(i,j) \in \mathcal{B}_1^{(k)}$ that $(k,i) \in \mathcal{B}_1^{(j)}$. Hence, job j meets our requirements since $p_j > 0$ and $\mathcal{B}_1^{(j)} \neq \emptyset$. □

Proof of Lemma 5(c). Let $\mathcal{B} = \mathcal{B}_s^{(k)}$, for any set $\mathcal{B}_s^{(k)} \in \mathfrak{B}$. We have to distinguish two cases, depending on whether the oriented 3-cycle contains k or not.

Case (1): For an oriented 3-cycle $\langle ijk\rangle$ containing k, it cannot happen that $\alpha^{\mathcal{B}}_{\langle ijk\rangle} > \alpha_{\langle ijk\rangle}$, since this would require $\delta_{ij}+\delta_{jk}+\delta_{ki} \geq 2$ as well as $\delta_{ji}+\delta_{kj}+\delta_{ik} \geq 2$. This cannot hold at the same time by Constraints (2). Moreover, for any $(i,j) \in \mathcal{B}^{(k)}_s$ we have $\alpha^{\mathcal{B}}_{\langle ijk\rangle} = \alpha_{\langle ijk\rangle} - \Delta$, since $(j,i) \notin \mathcal{B}^{(k)}_s$ by Corollary 7.

Case (2): Now consider any 3-cycle $\langle ij\ell\rangle$ with $k \notin \{i,j,\ell\}$ (see Figure 4). The claim follows by showing that $\alpha^{\mathcal{B}}_{\langle ij\ell\rangle} \leq \alpha_{\langle ij\ell\rangle}$. We prove this by contradiction.

In order to have $\alpha^{\mathcal{B}}_{\langle ij\ell\rangle} > \alpha_{\langle ij\ell\rangle}$ the value of at least one variable from $\{\delta_{ij}, \delta_{j\ell}, \delta_{\ell i}\}$ have to increase during Transformation (11), which in turn means to decrease some from $\{\delta_{ji}, \delta_{\ell j}, \delta_{i\ell}\}$. Note that by Constraints (2) it holds

$$(\delta_{ji}+\delta_{ik}+\delta_{kj}) + (\delta_{\ell j}+\delta_{jk}+\delta_{k\ell}) + (\delta_{i\ell}+\delta_{\ell k}+\delta_{ki}) + (\delta_{ij}+\delta_{j\ell}+\delta_{\ell i}) = 6. \quad (15)$$

Now consider the set $\mathcal{D} := \{(\delta_{ji}+\delta_{ik}+\delta_{kj}), (\delta_{\ell j}+\delta_{jk}+\delta_{k\ell}), (\delta_{i\ell}+\delta_{\ell k}+\delta_{ki})\}$. It is easy to see that not all values in $\mathcal{D}$ can be larger than 2, otherwise this contradicts Equation (15). Moreover, by Transformation (11), in order to have $\alpha^{\mathcal{B}}_{\langle ij\ell\rangle} > \alpha_{\langle ij\ell\rangle}$, at least one value in $\mathcal{D}$ should be larger than 2. By these observations, we have to examine the following two complementary cases.

In case (1), assume that only one of the three values in $\mathcal{D}$ is larger than 2. In this setting, the value of only one variable in $\{\delta_{ij}, \delta_{j\ell}, \delta_{\ell i}\}$ is increased after Transformation (11), and therefore we need $\delta_{ij}+\delta_{j\ell}+\delta_{\ell i} \geq 2$ in order to have $\alpha^{\mathcal{B}}_{\langle ij\ell\rangle} > \alpha_{\langle ij\ell\rangle}$. Moreover, it is necessary that the other two variables in $\{\delta_{ij}, \delta_{j\ell}, \delta_{\ell i}\}$ do not decrease, which in turn means that the other two values in $\mathcal{D}$ are at least 1. All these conditions together contradict Equation (15).

In case (2), we assume that at least two values in $\mathcal{D}$ are larger than 2. To increase $\alpha_{\langle ij\ell\rangle}$, one needs $\delta_{ij}+\delta_{j\ell}+\delta_{\ell i} \geq 2-\Delta$ to begin with, which again contradicts Equation (15). $\square$

7 Efficient Implementation of Transformation

In this section we show how to implement the transformation described in Theorem 4, which turns a solution for [CH-Δ] into a feasible solution for [P-Δ] without deteriorating the objective function value. The algorithm, which is called `Repair`, is sketched in Figure 5. The total running time required by `Repair` is shown to be bounded by $O(n^3)$.

Let us start by giving some explanations on the pseudo-code displayed in Figure 5. In the proof of Lemma 5 we have seen that if the current solution is not feasible for [P-Δ], then there exists at least a nonempty set $\mathcal{B}$ for which $V(\mathcal{B}) \leq 0$ (see Formula (12)). Then, we pick up this $\mathcal{B}$, and apply Transformation (11) during the for-loop $(6-15)$. After Transformation (11) has been completed, the same steps as before are repeated with the new solution, until the condition at Step 5 is not satisfied any more, i.e., until the solution is feasible for [P-Δ].

Note that, as soon as any δ_{ij} decreases at Steps 7, then we may need to update all $\mathcal{B}^{(k)}_s$ and $V\left(\mathcal{B}^{(k)}_s\right)$ with $(i,j) \in \mathcal{B}^{(k)}_s$. This task is performed at Steps 11 and 12. Observe that when δ_{ji} increases at Steps 8, then we do not need to update $\mathcal{B}^{(k)}_s$ and $V\left(\mathcal{B}^{(k)}_s\right)$, for $(j,i) \in \mathcal{B}^{(k)}_s$. Indeed, by the proof of Lemma 5(c) we know that no $\alpha_{\langle jik\rangle}$ can increase during Transformation (11). Therefore, if at a given point of `Repair` we have $(j,i) \notin \mathcal{B}^{(k)}_s$, then it cannot

```
Require: $\delta$ is a feasible solution for [CH-$\Delta$]
1: for all $k \in N$ and $1 \le s \le 1/\Delta$ do
2:     Compute $\mathcal{B}_s^{(k)} := \{(i,j) : \alpha_{\langle ijk \rangle} \ge s\Delta\}$
3:     Compute $V\left(\mathcal{B}_s^{(k)}\right) := \sum_{(i,j)\in\mathcal{B}_s^{(k)}} p_j w_i - \sum_{(i,j)\in\mathcal{B}_s^{(k)}} p_i w_j$
4: end for
5: while there is a set $\mathcal{B}$ such that $\mathcal{B} \neq \emptyset$ and $V(\mathcal{B}) \le 0$ do
6:     for all $(i,j) \in \mathcal{B}$ do
7:         $\delta_{ij} := \delta_{ij} - \Delta$
8:         $\delta_{ji} := \delta_{ji} + \Delta$
9:         for all $k \in N$ and $1 \le s \le 1/\Delta$ do
10:            if $(i,j) \in \mathcal{B}_s^{(k)}$ and $\alpha_{\langle ijk \rangle} < s\Delta$ then
11:                $V\left(\mathcal{B}_s^{(k)}\right) := V\left(\mathcal{B}_s^{(k)}\right) - (p_j w_i - p_i w_j)$
12:                Remove $(i,j)$ from $\mathcal{B}_s^{(k)}$
13:            end if
14:        end for
15:    end for
16: end while
```

Figure 5: Algorithm `Repair`.

happen any more in the following steps of `Repair` that $(j,i) \in \mathcal{B}_s^{(k)}$. By these arguments, it easily follows that increasing δ_{ji} does not change any $\mathcal{B}_s^{(k)}$ and $V\left(\mathcal{B}_s^{(k)}\right)$.

Theorem 10. *Algorithm* `Repair` *runs in* $O(n^3)$ *time.*

Proof. The claim on the total running time required by `Repair`, can be explained by using the following observations. For any $k \in N$ and $1 \le s \le 1/\Delta$, the computation of $\mathcal{B}_s^{(k)}$ and $V\left(\mathcal{B}_s^{(k)}\right)$ takes $O(n^2)$ time. Therefore, the for-loop $(1-4)$ can be completed in $O(n^3)$ time. We assume a data structure with a bidirectional pointer between (i,j) and $\mathcal{B}$ whenever $(i,j) \in \mathcal{B}$. By exploiting this data structure, checking if $(i,j) \in \mathcal{B}$, or adding/removing (i,j) from set $\mathcal{B}$ are tasks that can be performed in $O(1)$ time. It is easy to see that the assumed data structure can be also computed within $O(n^3)$ time.

The claim on the running time follows by observing that any δ_{ij} can decrease at Steps (7) at most $1/\Delta$ times during the overall execution of `Repair` (although any δ_{ij} may also increase at some Step 8). To explain this, it is sufficient to observe that as soon as δ_{ij} decreases by Δ, then in the proof of Lemma 5(c) we have seen that any $\alpha_{\langle ijk \rangle}$ such that $(i,j) \in \mathcal{B}$, decreases by the same amount; Moreover, any $\alpha_{\langle ij\ell \rangle}$ for all $\langle ij\ell \rangle \in \mathcal{C}$, cannot increase after the application of Transformation (11). Since the value of any $\alpha_{\langle ijk \rangle}$ is not larger than 1, after at most $1/\Delta$ times δ_{ij} decreases, there are no more sets $\mathcal{B}$ to which pair (i,j) belongs, and the condition $(i,j) \in \mathcal{B}$ of the for-loop $(6-15)$ will not be satisfied any more. It follows that the for-loop (6-15) can be executed at most $\frac{n(n-1)}{2\Delta}$ times, as there are $n(n-1)/2$ different pairs of jobs. By the same amount we can clearly bound the number of times the while-loop (5-16) is executed.

Checking the while-condition at Step 5 is a task that can be performed in $O(n)$ time, since the number of sets $\mathcal{B}$ is bounded by n/Δ. Finally, it is easy to see that the for-loop (9-14) takes $O(n)$ time. It follows that the total running time spent to complete the while-loop (5-16) can be bounded by $O(n^3)$, and the claim follows. □

8 Open Problems

It would be interesting to investigate further connections between the vertex cover problem and $1|prec|\sum w_jC_j$. It has been proven that vertex cover cannot be approximated in polynomial time within a factor of 7/6, unless P=NP (Håstad [12]). Dinur & Safra [8] improved this bound to $10\sqrt{5}-21 \approx 1.36067$. It would be nice to have a similar result for $1|prec|\sum w_jC_j$ or, as suggested in [26], to prove that a polynomial time ρ-approximation algorithm for $1|prec|\sum w_jC_j$ implies the existence of a polynomial time ρ-approximation algorithm for the vertex cover problem.

Acknowledgments. The authors thank Andreas Schulz and an anonymous referee for useful comments. The first author is supported by Nuffield Foundation Grant NAL32608. The second author is supported by Swiss National Science Foundation project 200021-104017/1, “Power Aware Computing”, and by the Swiss National Science Foundation project 200020-109854, “Approximation Algorithms for Machine scheduling Through Theory and Experiments II”. The second author would like to dedicate this work to Emma on the occasion of her birth.

References

[1] C. Ambühl, M. Mastrolilli, N. Mutsanas, and O. Svensson. Scheduling with precedence constraints of low fractional dimension. In *Proceedings of 12th International Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 130–144, 2007.

[2] C. Ambühl, M. Mastrolilli, and O. Svensson. Approximating precedence-constrained single machine scheduling by coloring. In *Proceedings of 9th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 15–26, 2006.

[3] C. Ambühl, M. Mastrolilli, and O. Svensson. Inapproximability results for sparsest cut, optimal linear arrangement, and precedence constrained scheduling. In *Proceedings of 48th Annual Symposium on Foundations of Computer Science (FOCS)*, 2007. (To appear).

[4] K. A. Baker, P. C. Fishburn, and F. S. Roberts. Partial orders of dimension 2. *Networks*, 2:11–28, 1971.

[5] C. Chekuri and R. Motwani. Precedence constrained scheduling to minimize sum of weighted completion times on a single machine. *Discrete Applied Mathematics*, 98(1-2):29–38, 1999.

[6] F. A. Chudak and D. S. Hochbaum. A half-integral linear programming relaxation for scheduling precedence-constrained jobs on a single machine. *Operations Research Letters*, 25:199–204, 1999.

[7] J. R. Correa and A. S. Schulz. Single machine scheduling with precedence constraints. *Mathematics of Operations Research*, 30:1005–1021, 2005.

[8] I. Dinur and S. Safra. On the hardness of approximating minimum vertex cover. *Ann. of Math. (2)*, 162(1):439–485, 2005.

[9] M. X. Goemans and D. P. Williamson. Two-dimensional Gantt charts and a scheduling algorithm of Lawler. *SIAM J. Discrete Math.*, 13(3):281–294, 2000.

[10] R. Graham, E. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: A survey. In *Annals of Discrete Mathematics*, volume 5, pages 287–326. North–Holland, 1979.

[11] L. A. Hall, A. S. Schulz, D. B. Shmoys, and J. Wein. Scheduling to minimize average completion time: off-line and on-line algorithms. *Mathematics of Operations Research*, 22:513–544, 1997.

[12] J. Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859 (electronic), 2001.

[13] D. S. Hochbaum. Efficient bounds for the stable set, vertex cover and set packing problems. *Discrete Applied Mathematics*, 6:243–254, 1983.

[14] S. G. Kolliopoulos and G. Steiner. Partially-ordered knapsack and applications to scheduling. In *Proceedings of the 10th Annual European Symposium on Algorithms (ESA)*, pages 612–624, 2002.

[15] E. L. Lawler. Sequencing jobs to minimize total weighted completion time subject to precedence constraints. *Annals of Discrete Mathematics*, 2:75–90, 1978.

[16] E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, and D. B. Shmoys. Sequencing and scheduling: Algorithms and complexity. In S. C. Graves, A. H. G. R. Kan, and P. Zipkin, editors, *Handbooks in Operations Research and Management Science*, volume 4, pages 445–552. North-Holland, 1993.

[17] J. K. Lenstra and A. H. G. Rinnooy Kan. The complexity of scheduling under precedence constraints. *Operations Research*, 26:22–35, 1978.

[18] F. Margot, M. Queyranne, and Y. Wang. Decompositions, network flows and a precedence constrained single machine scheduling problem. *Operations Research*, 51(6):981–992, 2003.

[19] R. H. Möhring. Computationally tractable classes of ordered sets. In I. Rival, editor, *Algorithms and Order*, pages 105–193. Kluwer Academic, 1989.

[20] G. L. Nemhauser and L. E. Trotter. Properties of vertex packing and independence system polyhedra. *Mathematical Programming*, 6:48–61, 1973.

[21] G. L. Nemhauser and L. E. Trotter. Vertex packings: Structural properties and algorithms. *Mathematical Programming*, 8:232–248, 1975.

[22] V. T. Paschos. A survey of approximately optimal solutions to some covering and packing problems. *ACM Computing Surveys*, 29(2):171–209, 1997.

[23] N. N. Pisaruk. A fully combinatorial 2-approximation algorithm for precedence-constrained scheduling a single machine to minimize average weighted completion time. *Discrete Applied Mathematics*, 131(3):655–663, 2003.

[24] C. N. Potts. An algorithm for the single machine sequencing problem with precedence constraints. *Mathematical Programming Study*, 13:78–87, 1980.

[25] A. S. Schulz. Scheduling to minimize total weighted completion time: Performance guarantees of LP-based heuristics and lower bounds. In *Proceedings of the 5th Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 301–315, 1996.

[26] P. Schuurman and G. J. Woeginger. Polynomial time approximation algorithms for machine scheduling: ten open problems. *Journal of Scheduling*, 2(5):203–213, 1999.

[27] J. B. Sidney. Decomposition algorithms for single-machine sequencing with precedence relations and deferral costs. *Operations Research*, 23:283–298, 1975.

[28] W. E. Smith. Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3:59–66, 1956.

[29] G. Steiner. Single machine scheduling with precedence constraints of dimension 2. *Mathematics of Operations Research*, 9(2):248–259, 1984.

[30] W. T. Trotter. *Combinatorics and Partially Ordered Sets: Dimension Theory.* The John Hopkins University Press, Baltimore and London, 1992.

[31] G. J. Woeginger. On the approximability of average completion time scheduling under precedence constraints. *Discrete Applied Mathematics*, 131(1):237–252, 2003.