Partially-Ordered Knapsack and Applications to Scheduling

Stavros G. Kolliopoulos* George Steiner†

Abstract

In the *partially-ordered knapsack* problem (POK) we are given a set N of items and a partial order $\prec_P$ on N. Each item has a size and an associated weight. The objective is to pack a set $N' \subseteq N$ of maximum weight in a knapsack of bounded size. N' should be precedence-closed, i.e., be a valid prefix of $\prec_P$. POK is a natural generalization, for which very little is known, of the classical Knapsack problem. In this paper we present both positive and negative results. We give an FPTAS for the important case of a *2-dimensional* partial order, a class of partial orders which is a substantial generalization of the series-parallel class, and we identify the first non-trivial special case for which a polynomial-time algorithm exists. We also characterize cases where the natural linear relaxation for POK is useful for approximation but we demonstrate its limitations as well. Our results have implications for approximation algorithms for scheduling precedence-constrained jobs on a single machine to minimize the sum of weighted completion times, a problem closely related to POK.

1 Introduction

Let a partially-ordered set (poset) be denoted as $P = (N, \prec_P)$, where $N = \{1, 2, ..., n\}$. A subset $I \subseteq N$ is an *order ideal* (or *ideal* or *prefix*) of P if $b \in I$ and $a \prec_P b$ imply $a \in I$. In the *partially-ordered knapsack* problem (denoted POK), the input is a tuple $(P = (N, \prec_P), w, p, b)$ where P is a poset, $w : N \to R^+$, $p : N \to R^+$, and $b \in R^+$. For a set $S \subseteq N$, $p(S)$ ($w(S)$) denotes $\sum_{i \in S} p_i$ ($\sum_{i \in S} w_i$). We are given a knapsack of capacity b and the sought output is an ideal N' that maximizes $w(N')$ and fits in the knapsack, i.e., $p(N') \le b$. A ρ-approximation algorithm, $\rho < 1$, finds an ideal N' such that $p(N') \le b$ and $w(N')$ is at least ρ times the optimum. We occasionally abuse notation and denote a poset by N, or omit P from $\prec_P$ when this leads to no ambiguity. For simplicity we shall also sometimes denote a POK instance by N with $\prec_P, w, p$ implied from the context.

POK is a natural generalization of the classical Knapsack problem. An instance of the latter is a POK instance with an empty partial order. Johnson and Niemi [18] view POK as modeling, for example, an investment situation where every investment has a cost and a potential profit and in which certain investments can be made only if others have been made previously. POK is strongly NP-complete, even when $p_i = w_i$, $\forall i \in N$, and the partial order is bipartite [18] and hence does not have an FPTAS, unless $\mathcal{P} = \mathcal{NP}$. Very recently, Hajiaghayi et al. [14] showed that POK is hard to approximate within a factor $2^{(\log n)^\delta}$, for some $\delta > 0$, unless $3SAT \in DTIME(2^{n^{3/4+\varepsilon}})$.

*Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, Panepistimiopolis Ilissia, Athens 157 84, Greece (`www.di.uoa.gr/~sgk`). Part of this work was done while at the Department of Computing and Software at McMaster University, partially supported by NSERC grant 227809-00.

†Management Science and Information Systems, McMaster University, Hamilton, Ontario, L8S 4M4, Canada (`steiner@mcmaster.ca`). Research partially supported by NSERC Grant OG0001798.

The result relies on the hardness of bipartite clique by Feige and Kogan [11]. Using the hardness result of Khot [19] for the latter problem one obtains also that for any $\varepsilon > 0$, if SAT does not have a probabilistic algorithm that runs in $2^{n^{\varepsilon}}$ there is no polynomial time (possibly randomized) algorithm for POK that achieves an approximation ratio $n^{\varepsilon'}$ for some ε' that depends on ε. It is also worth noting that POK generalizes the well-studied densest k-subgraph problem (DkS). No NP-hardness of approximation result exists for DkS but the best approximation ratio currently known is $O(\min\{n^{\delta},\ n/k\})$, $\delta < 1/3$ [12, 1]. Feige [10] and Khot [19] have so far provided evidence that DkS may be hard to approximate within some constant factor.

In terms of positive results for POK, there is very little known. In 1983 Johnson and Niemi gave an FPTAS for the case when the precedence graph is a directed out-tree [18]. Recently there has been revived interest due to the relevance of POK for scheduling. An $O(1)$-factor approximation for POK would lead to a $(2-\beta)$-approximation, for some constant $\beta > 0$, for minimizing average completion time of precedence-constrained jobs on a single machine, a problem denoted as $1|prec|\sum w_j C_j$. Improving on the known factor of 2 for $1|prec|\sum w_j C_j$ (see, e.g., [4], [5], [15], [25], [28]) is one of the major open questions in scheduling theory [29]. The relationship between POK and $1|prec|\sum w_j C_j$ was explored in a recent paper by Woeginger [32].

Due to the scheduling connection, we adopt scheduling terminology for POK instances: items in N are *jobs,* function w assigns *weights* and function p *processing time.* To our knowledge, Woeginger gave after many years the first new results for POK by showing pseudopolynomial algorithms for the cases where the underlying partial order is an interval order or a bipartite convex order [32].

In this paper we present both positive and negative results for POK. Our positive results are based on structural information of posets. One of the main applications of posets is in scheduling problems but there are only a few relevant results (e.g., [6], [15]). Moreover, these results are usually derived either by simple greedy scheduling or by relying on an LP solution to resolve the ordering. However, a large amount of combinatorial theory exists for posets. Tapping this source can only help in designing approximation algorithms. Following this approach, we obtain combinatorial algorithms for comprehensive classes of POK instances. These lead to improved approximation algorithms for the corresponding cases of $1|prec|\sum w_j C_j$. Perhaps ironically, we then show that the natural LP relaxation for POK provides only limited information.

The first part of our paper deals with the complexity of POK on two classes of partial orders. First, we give an FPTAS for POK when the underlying order is 2-dimensional. Second, we give a polynomial-time algorithm for a special class of bipartite orders.

2-dimensional orders. In Section 2 we provide an FPTAS for POK when the underlying order is *2-dimensional.* It achieves a $(1-\varepsilon)$-approximation for the optimum weight while meeting the upper bound on the processing time. We proceed to give background on 2-dimensional orders. A *linear extension* of a poset $P = (N, \prec_P)$ is a linear (total) order L with $a \prec_P b$ implying $a \prec_L b$ for $a, b \in N$. Every poset P can be defined as the intersection of its linear extensions (as binary relations) [31]. The minimum number of linear extensions defining P in this way is the *dimension of* P, denoted by $dimP$. It is well known that $dimP = 2$ exactly when P can be embedded into the Euclidean plane so that $a \prec_P b$ for $a, b \in N$ if and only if the point corresponding to a is not to the right and not above the point corresponding to b. 2-dimensional posets were first characterized by Dushnik and Miller [9] and they can be recognized and their two defining linear extensions can be found in polynomial time. However, recognizing whether $dimP = k$ for any $k \geq 3$ is NP-complete [33]. POK is NP-complete on 2-dimensional partial orders, since the empty partial order is also of dimension 2. It is well known that every directed out-tree poset is series-parallel and that every series-parallel poset is also of dimension 2, but the class of 2-dimensional posets is substantially larger. For example, while the class of series-parallel posets can be characterized by a single forbidden subposet, posets of dimension 2 cannot be defined by a finite list of forbidden substructures. Thus our FPTAS for 2-dimensional POK represents a substantial addition to previously known positive results on directed out-trees [18] and other classes [32]. For a review of the extensive literature on 2-dimensional posets, we refer the reader to [26].

Complement of chordal bipartite orders. A *bipartite poset,* denoted as $(X, Y; \prec)$, is one whose comparability graph is a bipartite graph $G = (X, Y; E)$ with X, Y being the two sets of the vertex partition. By convention, the set of maximal elements of the partial order is Y. A POK instance is called *Red-Blue* if $\forall a \in N$, either $w_a = 0$ (a is red) or $p_a = 0$ (a is blue). Red-Blue bipartite instances of POK are of particular interest since solving any POK instance can be reduced in an approximation-preserving manner to solving a Red-Blue bipartite instance (cf. Sec. 3). Observe that on such an instance, we can assume without loss of generality, that the set of red elements is X. In Section 3 we give a polynomial-time algorithm for POK on Red-Blue bipartite instances where the comparability graph has the following property: its bipartite complement is chordal bipartite. *Chordal bipartite graphs* are bipartite graphs in which every cycle of length 6 or more has a chord. They form a large class of perfect graphs, containing, for example, convex and biconvex bipartite graphs, bipartite permutation graphs (the comparability graphs of bipartite posets of dimension 2), bipartite distance hereditary graphs and interval bigraphs and they can be recognized in polynomial time [27]. For an excellent overview of these graph classes, the reader is referred to [2]. To the best of our knowledge, our result identifies the first nontrivial class of partial orders for which POK is solvable in polynomial time. The class we study is admittedly a restricted one but this is unavoidable to a certain extent: all other solvable cases, e.g., when the poset is a rooted tree [18], an interval order [32], a convex bipartite poset [32] or a series-parallel poset, include the case when the partial order is empty, i.e., the classical Knapsack problem. Hence their best algorithm can only be pseudopolynomial, unless $\mathcal{P} = \mathcal{NP}$. In contrast, the class we define does not include Knapsack; moreover it is the "maximal" possible in $\mathcal{P}$ since without the Red-Blue constraint the problem becomes NP-hard even on these restricted posets. We also give an FPTAS for POK with general w and p functions and comparability graph whose bipartite complement is chordal bipartite.

As a corollary to our POK results, we obtain in Section 4 an 1.61803-approximation for $1|prec|\sum w_j C_j$ when the partial order of the jobs falls in one of the two classes we described above. Our derivation uses machinery developed by Woeginger in [32].

In the second part of our paper we turn our attention to the problem with a general partial order. We study the natural linear relaxation and provide insights on the structure of optimal solutions. On the positive side we provide in Section 5 a very simple bicriteria-type approximation for weight and processing time on the inputs that meet the so-called weight-majority condition (see Section 5 for details). On the negative side, the LP solution is inadequate in the general case. Let the *rank* of an ideal I be defined as $w(I)/p(I)$. Poset N is *indecomposable* if the only maximum-rank ideal is the entire set. It is known that in order to improve on the 2-approximation for $1|prec|\sum w_j C_j$ one needs only to consider indecomposable instances [4, 30]. As part of our contribution, we show that indecomposability affects POK, although in a different manner. In Section 6 we show that if the input is indecomposable the LP-relaxation provides essentially no information since the optimal solution returns the same fraction of $p(N)$ and $w(N)$ (cf. Theorem 6.1).

Our guarantee for the weight in the bicriteria-type result is not an approximation ratio in the classical sense (cf. Lemma 5.2) and as said applies only for the weight-majority case. We give evidence in Section 7 that both these limitations of our algorithm reflect rather the difficulty of the problem itself. We show that an $O(1)$-approximation algorithm for the POK problem restricted to weight-majority instances, would imply a breakthrough $(2-\beta)$-approximation for $1|prec|\sum w_jC_j$, $\beta > 0$. This is a rather surprising connection given that any weight-majority instance defined on an indecomposable poset is by definition infeasible. As mentioned, the hard case for $1|prec|\sum w_jC_j$ is precisely the one where the underlying poset is indecomposable.

In summary, our main contributions in this paper are an FPTAS for POK on 2-dimensional orders and the first non-trivial polynomially solvable case. We also formalize the insight that the natural linear program is inadequate in the general case by examining the role of indecomposable partial orders.

The outline of the paper is as follows. In Section 2 we examine POK on 2-dimensional orders. In Section 3 we present the polynomially solvable case. In Section 4 we investigate the applications to the approximability of the scheduling problem. In Sections 5 and 6 we study the linear programming relaxation for POK with general precedence constraints. Finally, in Section 7 we establish the connection between the approximability of weight-majority instances of POK and $1|prec|\sum w_jC_j$.

An extended abstract of the present paper appeared in [22].

2 POK on 2-dimensional orders

2.1 A pseudopolynomial algorithm for POK on 2-dimensional posets

Consider a POK instance $(P=(N,\prec_P),p,w,b)$ where $\prec_P$ is 2-dimensional. For simplicity we denote $w(N)$ by W. Without loss of generality we assume that processing times, weights and the knapsack capacity are all integers.

For any $S\subseteq N$ and $w\in[0,W]$, let $p_w(S)=\min\{p(I)|I\subseteq S$ is an ideal in the induced poset S and $w(I)=w\}$. Note that if there is no ideal in S with weight w then $p_w(S)=+\infty$ by default. For $k\in S$ and $w\in[0,W]$, define further $p_w(S,k)=\min\{p(I)|I\subseteq S$ is an ideal in the induced poset S, the highest numbered element in I is k and $w(I)=w\}$. If there is no ideal in S with weight w and containing k as its highest numbered element then $p_w(S,k)=+\infty$. It is clear that $p_w(S)=\min_{k=1,2,\ldots,n}p_w(S,k)$.

We assume, without loss of generality, that the elements of N have been numbered so that $L_1=1,2,...,n$ is a linear extension of P. We will use the notation $i||k$ if $i<k$ as numbers and i is not comparable to k in P. The *principal ideals* are defined by $B_k=\{i\mid i\preceq_P k\}$ and their *'complements'* by $\overline{B}_k=\{i\mid i||k\}$ for $k=1,2,...,n$. Partition the ideals of P by their highest numbered element, and let $\mathcal{I}_k$ be the set of ideals with highest numbered element k. If $I\in\mathcal{I}_k$, then we clearly must have $B_k\subseteq I$ and $I\backslash B_k$ must be an ideal in the induced subposet $\overline{B}_k$. Therefore, each ideal $I\in\mathcal{I}_k$ is in a one-to-one correspondence with the ideal $I\backslash B_k$ of the subposet $\overline{B}_k$ and $p(I)=p(I\backslash B_k)+p(B_k)$ and $w(I)=w(I\backslash B_k)+w(B_k)$. Thus, we have proved the following lemma.

Lemma 2.1 *Consider a POK problem on the poset P with knapsack capacity b and total item weight W. Then for any weight $w\in[w(B_k),W]$, $p_w(N,k)=p_{w-w(B_k)}(\overline{B}_k)+p(B_k)$.*

In order to derive $p_w(N)$, we could apply the computation in Lemma 2.1 recursively to the subposets $\overline{B}_k$, however, this would yield a computation, which is exponential in n. As the following theorem shows, however, the recursion does not need to go beyond the second level if $\dim P=2$, thus yielding a pseudopolynomial algorithm. For the remainder of the section, let us assume that $L_1 = 1, 2, ..., n$ and L_2 are the two defining linear extensions for the 2-dimensional poset P. Accordingly $i \prec_P k$ iff $i \prec_{L_1} k$ and $i \prec_{L_2} k$.

Theorem 2.1 *If $dimP = 2$ for a POK problem with knapsack capacity b and total item weight W, then its optimal solution can be computed by a pseudopolynomial algorithm in $O(n^2W)$ time.*

Proof. Partition the set of ideals in P by their highest numbered element and apply Lemma 2.1. Accordingly we obtain $p_w(N)$ by

$$p_w(N) = \min_{k=1,2,...,n} p_w(N, k) = \min_{k=1,2,...,n} \{p_{w-w(B_k)}(\overline{B}_k) + p(B_k) | w - w(B_k) \geq 0\} \text{ for } 0 \leq w \leq W.$$

We will show that the computation of $p_x(\overline{B}_k)$ for a fixed k and all $x \in [0, W]$ can be carried out in $O(nW)$ time. Then the entire computation for the $p_w(N)$ needs no more than $O(n^2W)$ time. Let

$$C_{kj} = \overline{B}_k \cap B_j = \{i \mid \ i \in \overline{B}_k, i \preceq_P j\} \text{ and}$$
$$\overline{C}_{kj} = \overline{B}_k \cap \overline{B}_j = \{i \mid \ i \in \overline{B}_k, i < j, i \not\prec_P j\} \text{ for } k = 1, 2, ..., n \ \text{ and } j||k.$$

See Fig. 1 for an illustration of the various sets. We claim that $dimP = 2$ and $j||k$ imply $\overline{C}_{kj} = \overline{B}_j$: If $i \in \overline{C}_{kj}$, then it can easily be seen that $i \in \overline{B}_j$. For the other direction, if $i \in \overline{B}_j$, then $i < j$ and $i \not\prec_P j$, i.e., $j \prec_{L_2} i$. Furthermore, $j||k$ implies $j < k$ and $j \not\prec_P k$, i.e., $k \prec_{L_2} j$ too, so that by transitivity of L_2 $k \prec_{L_2} i$ also holds, which implies $i \in \overline{B}_k$, and thus $i \in \overline{C}_{kj}$.

Let us compute the $p_x(\overline{B}_k)$ in ascending order of $k = 1, 2, ..., n$. For this it suffices to calculate $p_x(\overline{B}_k, j)$ for all $j : j||k$ and $x \in [0, W]$, since $p_x(\overline{B}_k) = \min_{j:j||k}\{p_x(\overline{B}_k, j)\}$. Applying Lemma 2.1 to the poset $\overline{B}_k$ shows that $p_x(\overline{B}_k, j)$ can be derived from $p_x(\overline{C}_{kj})$ by $p_x(\overline{B}_k, j) = p_{x-w(C_{kj})}(\overline{C}_{kj}) + p(C_{kj})$ for $w(C_{kj}) \leq x \leq W$. But $\overline{C}_{kj} = \overline{B}_j$ and $p_{x-w(C_{kj})}(\overline{C}_{kj}) = p_{x-w(C_{kj})}(\overline{B}_j)$, therefore we need only previously computed p values for $\overline{B}_j$. Combining these observations, we obtain

$$p_x(\overline{B}_k) = \min_{j:j||k}\{p_x(\overline{B}_k, j)\} = \min_{j:j||k}\{p_{x-w(C_{kj})}(\overline{B}_j) + p(C_{kj}) | x - w(C_{kj}) \geq 0\}.$$

Finally, to obtain the optimal solution for the POK we only have to find w^*, the largest weight for which $p_{w^*}(N) \leq b$.

Since the quantities $p(B_k), w(B_k), p(C_{kj})$ and $w(C_{kj})$ can all be calculated in $O(n^3)$ time in a preprocessing step, the claimed complexity of the algorithm easily follows. ∎

It is also possible to define the 'dual' of the above procedure: For any $S \subseteq N$ and $p \in [0, b]$, let $w_p(S) = \max\{w(I) | I \subseteq S$ is an ideal in the induced poset S and $p(I) = p\}$. Note that if there is no ideal in S with total processing time p then we define $w_p(S) = -\infty$. For $k \in S$ and $p \in [0, b]$, define further $w_p(S, k) = \max\{w(I) | I \subseteq S$ is an ideal in the induced poset S, the highest numbered element in I is k and $p(I) = p\}$. If there is no ideal in S with total processing time p and containing k as its highest numbered element then $w_p(S, k) = -\infty$. It is clear that $w_p(S) = \max_{1,2,...,k} w_p(S, k)$. We can set up analogous recursions to the ones above to obtain an alternative pseudopolynomial solution for any 2-dimensional instance of POK. We omit the details and state only the result that follows from this.

Theorem 2.2 *If $dimP = 2$ for a POK problem with knapsack capacity b and total item value W, then its optimal solution can be computed by a pseudopolynomial algorithm in $O(n^2b)$ time.*

By Theorems 2.1 and 2.2 interesting cases, such as 2-dimensional instances where all jobs have unit weights or unit processing times, can be solved in polynomial time.

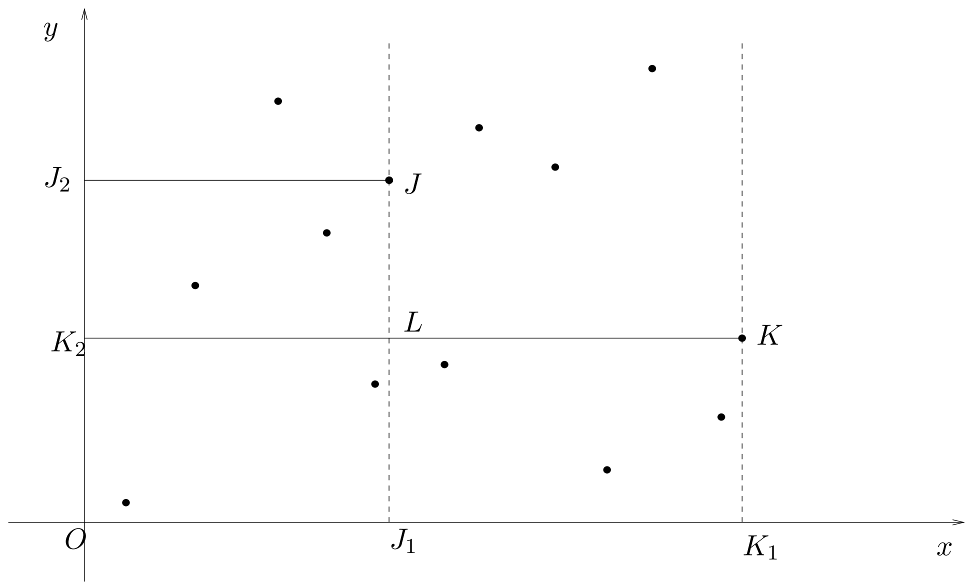

Figure 1: Sets defined by the 2-d algorithm. Linear extension L_1 corresponds to a nondecreasing order of the x-coordinates of the jobs. A set is *defined by an area* R means the set contains the points in R. Jobs k, j correspond to points K, J respectively. The sets B_k, B_j are defined by rectangles OK_2KK_1 and OJ_2JJ_1 respectively. Set $\bar{B}_k$ is defined by the half-open area above K_2K, to the right of the y-axis and to the left of the line $x = K_1$. Set $\bar{B}_j$ is defined by the half-open area above J_2J, to the right of the y-axis and to the left of the line $x = J_1$. Set C_{kj} is defined by rectangle K_2J_2JL. Only sets B_k, B_j and C_{kj} contain the borders of their defining areas.

2.2 Obtaining an FPTAS

The algorithm given in Theorem 2.1 is polynomial in W. We show that compressing the data in a standard manner allows us to compute a near-optimal solution instead of the exact optimum in time polynomial in n and $1/\varepsilon$, for a prescribed error bound ε.

We will scale the coefficients in a way similar to the one used for Knapsack (cf. [16]). Let $K = \varepsilon w_{\max}/n$ be the scaling parameter, where $w_{\max} = \max_{j\in N} w_j$. Set $w'_j = \lfloor w_j/K \rfloor \ \forall j \in N$.

Let $I(K)$ and I^* be the optimal ideals for the scaled and the original problem, respectively. Since any job j for which $\sum_{i\preceq_P j} p_i > b$ can be eliminated together with all its successors in P in a preprocessing step, we can assume without the loss of generality that every job fits into the knapsack, i.e., $\sum_{i\preceq_P j} p_i \le b$ for every $j \in N$. This implies that $w_j \le w(I^*)$ for every $j \in N$ and thus $w_{\max} \le w(I^*)$. Then we have

$$w(I^*) \ge w(I(K)) \ge Kw'(I(K)) \ge Kw'(I^*) \ge K\sum_{j\in I^*}(w_j/K - 1) = w(I^*) - K|I^*|.$$

This implies that the relative error satisfies

$$(w(I^*) - w(I(K))/w(I^*) \le K|I^*|/w(I^*) \le \varepsilon w_{\max}|I^*|/nw(I^*) \le \varepsilon,$$

where the last inequality holds because $w_{\max} \le w(I^*)$.

Applying the algorithm of Theorem 2.1 to the scaled instance will require no more than $O(n^2 w'(N)) \le O(n^2 W/K) = O(n^4/\varepsilon)$ time. Thus we have proved

Theorem 2.3 *For a POK problem on poset $P = (N, \prec_P)$ with knapsack capacity b where $dimP = 2$ there is an FPTAS which for any $\varepsilon > 0$ produces in time $O(n^4/\varepsilon)$ an ideal of processing time at most b and weight at least $(1-\varepsilon)$ times the optimum.*

3 *POK* on special bipartite partial orders

POK is strongly NP-complete on bipartite orders [18]. In fact the hardness result of [14] holds for a bipartite Red-Blue instance. Moreover the following holds:

Lemma 3.1 *There is an approximation-preserving reduction of a POK problem on a poset $P = (N_0, \prec_P)$ to a bipartite Red-Blue POK instance.*

Proof. Given a general input N_0 one can transform it to a bipartite input by having for each job $i_0 \in N_0$ two vertices i, i', one on each side X, Y of the partition. The vertex i on the X-side assumes the processing time and i' the weight of i_0. Furthermore i has zero weight and i' has zero processing time. Precedence constraint $i_0 \prec j_0$ for $i_0, j_0 \in N_0$ translates to $i \prec j'$. Moreover $i \prec i'$ for all i. Without loss of generality we can assume that in any ideal of N, inclusion of i implies inclusion of i'. Thus there is a one-to-one correspondence between ideals of N and N_0, with the total processing time and weight being the same. ∎

In contrast, we present an efficient algorithm for a special bipartite order. We show first some useful facts. Let $(X, Y; \prec)$ be a bipartite poset with comparability graph $G = (X, Y; E)$. Its bipartite complement $\overline{G} = (X, Y; \overline{E})$ is defined by $\overline{E} = X \times Y \backslash E$. An ideal $X' \cup Y'$, where $X' \subseteq X$ and $Y' \subseteq Y$, is *Y-maximal* if there is no $y \in Y \setminus Y'$ such that $X' \cup Y' \cup \{y\}$ is also an ideal. The ideal (X', Y') is *X-minimal* if there is no $x \in X'$ such that $(X'\backslash x, Y')$ is also an ideal.

Each ideal I of a poset is uniquely defined by its *maximal elements* $\max I = \{a \in I \mid \nexists b \in I$ such that $a \prec b\}$: If we know $\max I$, then $I = \{a \mid \exists b \in \max I$ such that $a \preceq b\}$. The elements of $\max I$ always form an antichain in the poset, and an antichain generates an ideal in this sense. Let $\mathcal{M} = \{I \mid I$ is an ideal such that $\max I$ is a maximal antichain$\}$. It is well known that $(\mathcal{M}, \subseteq)$ is a lattice which is isomorphic to the lattice of maximal antichains.

Lemma 3.2 *The following statements are equivalent in a bipartite poset $(X, Y; \prec)$ with comparability graph $G = (X, Y; E)$:*

1. *$X' \cup Y'$ is a Y-maximal and X-minimal ideal.*
2. *$Y' \cup (X \backslash X')$ is a maximal antichain and $(Y' \cup X) \in \mathcal{M}$.*
3. *The induced subgraph $\overline{G}[X \backslash X', Y']$ is a maximal complete bipartite subgraph of the bipartite complement $\overline{G}$.*

Proof. 1.$\Longrightarrow$ 2. Suppose $X' \cup Y'$ is a Y-maximal and X-minimal ideal. We cannot have any edge between $X \backslash X'$ and Y' in G, since (X', Y') is an ideal. Thus $Y' \cup (X \backslash X')$ is an antichain. Since Y' is Y-maximal, every $y \in Y \backslash Y'$ must have a predecessor in $X \backslash X'$, so there is no $y \in Y \backslash Y'$ such that $Y' \cup (X \backslash X') \cup y$ would also be an antichain. Similarly, every $x \in X'$ must have a successor $y \in Y'$, since (X', Y') is X-minimal, so there is no $x \in X'$ for which $Y' \cup (X \backslash X') \cup x$ would also be an antichain. This proves the maximality of the antichain $Y' \cup (X \backslash X')$. Accordingly, the ideal $(Y' \cup X)$, which is generated by $Y' \cup (X \backslash X')$, is in $\mathcal{M}$.

2.$\Longrightarrow$ 3. obvious.

3.$\Longrightarrow$ 1. Let $\overline{G}[X \backslash X', Y']$ be a maximal complete bipartite subgraph of $\overline{G}$. This implies that no $x \in X \backslash X'$ can be a predecessor for any $y \in Y'$, thus (X', Y') is an ideal in $(X, Y; \prec)$. Since $\overline{G}[X \backslash X', Y']$ is a maximal complete bipartite subgraph of $\overline{G}$, no $y \in Y \backslash Y'$ is connected to every element of $X \backslash X'$ in $\overline{G}$, so there is no $y \in Y \backslash Y'$ extending the ideal $X' \cup Y'$, i.e., it is Y-maximal. Similarly, no $x \in X'$ is connected to every element of Y' in $\overline{G}$, so $X' \cup Y'$ is also X-minimal. ∎

It is clear that in order to solve a Red-Blue bipartite instance of POK, we need to search through only the ideals which are Y-maximal and X-minimal. By the above lemma, these ideals of a bipartite poset $(X, Y; \prec)$ are in a one-to-one correspondence with the maximal complete bipartite subgraphs of the bipartite complement of its comparability graph. If we have such a subgraph $\overline{G}[U, Z]$, then the corresponding ideal $X \backslash U \cup Z$, its weight $w(X \backslash U \cup Z)$ and processing time $p(X \backslash U \cup Z)$ can all be computed in $O(n)$ time. Furthermore, if the bipartite complement $\overline{G}$ is chordal bipartite, then it has only at most $|\overline{E}|$ maximal complete bipartite subgraphs [21]. Kloks and Kratsch [20] found an algorithm which lists these in $O(|X \cup Y| + |\overline{E}|)$ time if $\overline{G}$ is given by an appropriately ordered version of its bipartite adjacency matrix. This proves the following theorem.

Theorem 3.1 *Consider a* Red-Blue *instance of POK on a bipartite poset $(X, Y; \prec)$ with comparability graph G. If $\overline{G}$ is chordal bipartite then there is an algorithm which solves this POK problem in $O(n^3)$ time.*

We proceed now to lift the restriction that the bipartite instance is Red-Blue. The resulting problem is NP-hard, even on this restricted class of posets, since it includes as a special case the classical Knapsack problem. To see this, divide the Knapsack instance arbitrarily into two parts. This is an empty bipartite order whose bipartite complement is clearly chordal bipartite.

Let $I = B \cup C$ be an arbitrary ideal in the bipartite poset $(X, Y; \prec)$, where $B \subseteq X$ and $C \subseteq Y$. It is clear that $\max I$ partitions into $C \cup (\max I \cap B)$. Let $X' = B \backslash \max I$ and $Y' = \{y \in Y \mid \nexists x \in X \backslash X'$

such that $x \prec y\}$. It is clear that $C \subseteq Y'$ and that $X' \cup Y'$ is a Y-maximal and X-minimal ideal. Then by Lemma 3.2, $Y' \cup (X \backslash X')$ is a maximal antichain containing $\max I$. The ideal generated by this maximal antichain is $Y' \cup X \in \mathcal{M}$. Furthermore, $I \subseteq (Y' \cup X)$ and $(Y' \cup X) \backslash I$ is an antichain contained in $Y' \cup (X \backslash X')$. Thus I can be *derived* from $(X \cup Y')$ by the deletion of an appropriate unordered subset of $Y' \cup (X \backslash X')$. If we considered for deletion all such subsets of $Y' \cup (X \backslash X')$, and repeated this for all $(X \cup Y') \in \mathcal{M}$, then we would derive *every* ideal of the poset $(X, Y; \prec)$ (some of them possibly several times.)

Consider now an arbitrary instance of POK on a bipartite poset $(X, Y; \prec)$ with knapsack size b. Let $(X \cup Y') \in \mathcal{M}$. If $p(X \cup Y') \leq b$, then clearly we don't need to delete any antichain from $X \cup Y'$, the feasible ideal of maximum weight derivable from $X \cup Y'$ is $X \cup Y'$ itself. Otherwise, $X \cup Y'$ is infeasible, i.e., $p(X \cup Y') > b$. Define $X' = \{x \in X \mid \exists y \in Y'$ such that $x \prec y\}$. We can find the largest-weight feasible ideal derivable from $X \cup Y'$ by the above process by solving the auxiliary classical Knapsack problem $\{\max w(J) \mid J \subseteq ((X \backslash X') \cup Y')), p(J) \leq b - p(X')\}$. The optimal solution J^* of this can be found by a pseudopolynomial algorithm in $O(nW)$ time, and $X' \cup J^*$ is the largest-weight feasible ideal that can be derived from this $(X \cup Y')$ for the original POK instance. As we have discussed earlier, if $(X, Y; \prec)$ is a bipartite poset whose comparability graph has a bipartite complement $\overline{G} = (X, Y; \overline{E})$ which is chordal bipartite, then $|\mathcal{M}| \leq |\overline{E}|$, so that we need to call the pseudopolynomial algorithm for the solution of at most $|\overline{E}|$ auxiliary problems. This proves the following.

Theorem 3.2 *Consider an instance of POK on a bipartite poset $(X, Y; \prec)$ with comparability graph G and total item weight W. If $\overline{G}$ is chordal bipartite then there is a pseudopolynomial algorithm which solves this POK problem in $O(n^3 W)$ time.*

It is easy to see that invoking as a Knapsack oracle the FPTAS in [17], instead of a pseudopolynomial algorithm, yields a $(1 - \varepsilon)$ weight-approximation to the original problem.

Theorem 3.3 *Consider an instance of POK on a bipartite poset $(X, Y; \prec)$ with comparability graph G. If $\overline{G}$ is chordal bipartite then there is an FPTAS which for any $\varepsilon > 0$, solves this POK problem in $O(n^3 \log(1/\varepsilon) + n^2(1/\varepsilon^4))$ time.*

4 Applications to scheduling

In this section we show how the above pseudopolynomial algorithms for POK given in Theorems 2.1 and 3.2 lead to improved polynomial-time approximation algorithms for special cases of the scheduling problem $1|prec|\sum w_j C_j$. The following results of Woeginger [32] can be used in the transition between the two problems.

Lemma 4.1 *[32] If there is a polynomial-time ρ-approximation algorithm for the special case of $1|prec|\sum w_j C_j$ where $1 \leq w_j \leq n^2$ and $1 \leq p_j \leq n^2$ holds for all jobs j, then for every $\varepsilon > 0$, there exists a polynomial-time $(\rho + \varepsilon)$-approximation algorithm for the general problem $1|prec|\sum w_j C_j$.*

Woeginger [32] also defined the following auxiliary problem, which is a special case of POK.
PROBLEM: GOOD INITIAL SET (IDEAL)
INSTANCE: An instance of $1|prec|\sum w_j C_j$ with nonnegative integer processing times p_j and weights w_j, a real number γ with $0 < \gamma \leq 1/2$.
QUESTION: Is there an ideal I in the poset P representing the precedence constraints for which $p(I) \leq (1/2 + \gamma)p(N)$ and $w(I) \geq (1/2 - \gamma)w(N)$?

The following theorem shows the strong connection between the solvability of POK and the approximability of $1|prec|\sum w_jC_j$. Its derivation uses 2-dimensional Gantt charts as introduced in [13].

Theorem 4.1 *[32] If $\mathcal{C}$ is a class of partial orders on which GOOD INITIAL SET is solvable in pseudopolynomial time, then for any $\varepsilon > 0$, the restriction of the scheduling problem to $\mathcal{C}$, i.e., $1|prec, \mathcal{C}|\sum w_jC_j$ has a polynomial-time $(\Phi+\varepsilon)$-approximation algorithm, where $\Phi = 1/2(\sqrt{5}+1) \approx 1.61803$.*

In view of the preceding developments, Theorem 4.1 leads to the following.

Corollary 4.1 *For any $\varepsilon > 0$, $1|prec, dimP = 2|\sum w_jC_j$ has a polynomial-time $(\Phi+\varepsilon)$-approximation algorithm.*

Proof. In light of Sidney's results [30] and the discussion in the Introduction, it is sufficient to consider only an indecomposable instance of $1|prec, dimP = 2|\sum w_jC_j$. Lemma 4.1 can be applied to this, since besides scaling the job parameters, its proof uses only the reversal of the precedence constraints, and the reverse of a 2-dimensional poset is clearly 2-dimensional again. Thus it suffices to consider only instances of $1|prec, dimP = 2|\sum w_jC_j$ where $w_i \leq n^2$ and $p_i \leq n^2$. By theorem 2.1 the corresponding instance of GOOD INITIAL SET can be solved in $O(n^2W) = O(n^4)$ time. ∎

After the first publication of this work [22] an improved 3/2-approximation for $1|prec, dimP = 2|\sum w_jC_j$ was obtained by Correa and Schulz [8]. Similarly by Theorems 4.1 and 3.2 we obtain:

Corollary 4.2 *For any $\varepsilon > 0$, the problem $1|prec, \prec_P|\sum w_jC_j$ where P is a bipartite poset $(X, Y; \prec_P)$ such that the bipartite complement of its comparability graph is a chordal bipartite graph, has a polynomial-time $(\Phi+\varepsilon)$-approximation algorithm.*

5 The LP-relaxation for general POK

In this section we examine the natural integer program for POK and the associated linear relaxation. For the remainder of the paper we denote the knapsack capacity b as $p(N)/l$ where $l > 1$. Consider the following integer program with parameter l :

$$\text{maximize} \sum_{j\in N} w_jx_j$$

$$\sum_{i\in N} p_ix_i \leq p(N)/l \tag{1}$$

$$x_j \leq x_i \quad \text{for} \quad i \prec j \tag{2}$$

$$x_i \in \{0,1\} \quad \text{for} \quad i \in N \tag{3}$$

Relaxing the integrality constraints to $0 \leq x_i \leq 1$, $i \in N$, gives a linear relaxation which we denote by $LP(l)$. Let $\bar{x}$ be a solution to $LP(l)$ with objective value $\sum_{j\in N} w_j\bar{x}_j = w(N)/h$, where $h \geq 1$. We say that the solution satisfies *weight majority* if $h < l$, i.e. the solution packs in the knapsack a higher fraction of the total weight compared to the fraction of the total processing time. If $h > l$ the solution satisfies *time majority.* We show how to round a weight-majority solution to an integral one, while relaxing by a small factor the right hand side of the packing constraint. The guaranteed value of the objective will also be less than $w(N)/h$, therefore producing a bicriteria approximation. The algorithm itself is simple. Let $\alpha \in (0,1)$ be a parameter of our choice. We apply filtering [24] based on α. Define $N' = \{j \in N \mid \bar{x}_j \geq \alpha\}$.

Lemma 5.1 *N' is an ideal and $p(N') \leq \frac{p(N)}{\alpha l}$. Moreover, $w(N') \geq \frac{1-h\alpha}{h(1-\alpha)} w(N)$.*

Proof. Let $j \in N'$ and $i \in N$ such that $i \prec j$. By (2), $\bar{x}_i \geq \bar{x}_j \geq \alpha$ implying $i \in N'$. Hence N' is an ideal. By (1) in $LP(l)$, $\sum_{j\in N'} p_j \bar{x}_j \leq p(N)/l$, and $p(N') \leq (1/\alpha)\sum_{j\in N'} p_j\bar{x}_j \leq p(N)/(\alpha l)$.

We now examine the weight of N'. By the choice of N', $\bar{x}_k < \alpha$ for $k \in N - N'$. Thus $\sum_{k\in N-N'} w_k\bar{x}_k + \sum_{f\in N'} w_f\bar{x}_f = w(N)/h$ which gives $\alpha w(N-N') + w(N') \geq w(N)/h \Rightarrow \alpha w(N - N') + w(N) - w(N-N') \geq w(N)/h$. Therefore $w(N-N') \leq w(N)(1-1/h)\frac{1}{1-\alpha}$ which implies $w(N') \geq \frac{1-h\alpha}{h(1-\alpha)} w(N)$. ∎

To ensure that Lemma 5.1 gives non-trivial bounds we must restrict the range of α. To ensure $1/(\alpha l) < 1$ and $1 - h\alpha > 0$, we require $\alpha > 1/l$ and $\alpha < 1/h$ respectively. We have shown the following lemma.

Lemma 5.2 *Let a POK instance be such that the linear relaxation $LP(l)$ has a solution $\bar{x}$ of value $w(N)/h$ with $1/l < 1/h$. For any $\alpha \in (1/l, 1/h)$ define $N' = \{j \in N \mid \bar{x}_j \geq \alpha\}$. N' is an ideal with the properties $w(N') \geq \frac{1-h\alpha}{h(1-\alpha)} w(N)$ and $p(N') \leq \frac{p(N)}{\alpha l}$.*

The given bounds show that to make $p(N')$ small, α needs to be large, while a small α gives a better weight guarantee. Accordingly the larger the gap $1/h - 1/l$, the more leeway we have for setting α far from both values.

Lemma 5.2 raises two questions. First, when can we expect the optimal fractional solution to meet the weight-majority condition $1/l < 1/h$? Second, the guarantee on the weight does not conform to the standard definition of a multiplicative ρ-approximation. Can we obtain a proper ρ-approximation on the weight with or without relaxing the upper bound on the processing time? We discuss these two questions in the next two sections.

6 The effect of decomposability

In this section we address the question: when can one hope to obtain a weight-majority solution to $LP(l)$ and hence apply Lemma 5.2? We show that decomposability of the input is a necessary condition by gaining insight into the structure of LP solutions. Finally we show an upper bound on the optimum of $LP(l)$ for general inputs.

To simplify our derivation we assume the input instance is bipartite Red-Blue. By Lemma 3.1 this is without loss of generality. For any $l > 1$ formulation $IPOK$, defined below, is the integer linear programming formulation of the associated bipartite Red-Blue POK instance.

$$\text{maximize} \sum_{j\in Y} w_j x_j$$

$$\sum_{i\in X} p_i x_i \leq p(X)/l \tag{4}$$

$$x_j - x_i \leq 0 \quad \text{for} \quad i \prec j \tag{5}$$

$$x_i \in \{0,1\} \quad \text{for} \quad i \in X \cup Y \tag{6}$$

In the linear programming relaxation, $LPOK$, the integrality constraints are relaxed to $0 \leq x_i \leq 1$ for $i \in X \cup Y$. We start with a simple lemma that lowerbounds the integrality gap of the formulation.

Lemma 6.1 *The integrality gap of $LPOK$ is at least $w(N)/l$.*

Proof. Consider a bipartite Red-Blue instance defined on a poset $(X, Y; \prec)$ with the following properties. Let $y_0 \in Y$ be a designated special job. For every job y in $Y - \{y_0\}$, let the set of predecessors of y be X. The only predecessor of y_0 is a single job x_0 in X with processing time less than or equal to $p(N)/l$. The weight of y_0 is equal to 1. By the given properties, the optimal integral solution to $IPOK$ on this poset has value 1. The fractional solution that assigns $1/l$ to all the variables is feasible for $LPOK$. Therefore the optimum value of $LPOK$ is at least $w(N)/l$. ∎

Observe that the instance defined in the proof of Lemma 6.1 is indecomposable. In the upcoming theorem we show a stronger result: an instance is indecomposable if and only if the optimum of LPOK is equal to $w(N)/l$.

Theorem 6.1 *The solution $\overline{x}_i = 1/l$ for $i \in X \cup Y$ is optimal for $LPOK$ with knapsack capacity $p(X)/l$ if and only if the underlying poset is indecomposable.*

Proof. The dual of $LPOK$, $DLPOK$, can be written as follows.

$$\text{minimize } \frac{p(X)}{l} u + \sum_{i \in X \cup Y} Z_i$$

$$p_i u + Z_i - \sum_{j : i \prec j} z_{ij} \geq 0 \quad \text{for } i \in X \tag{7}$$

$$Z_j + \sum_{i : i \prec j} z_{ij} \geq w_j \quad \text{for } j \in Y \tag{8}$$

$$u \geq 0 \tag{9}$$

$$Z_i \geq 0 \quad \text{for } i \in X \cup Y \tag{10}$$

$$z_{ij} \geq 0 \quad \text{for } i \prec j \tag{11}$$

It is clear that $\overline{x}_i = 1/l$ for $i \in X \cup Y$ is a feasible solution for $LPOK$. We call this solution *balanced* since it assigns the same value to variables from X and Y. Then the complementary dual solution of $DLPOK$ satisfies $\overline{Z_i} = 0$ for $i \in X \cup Y$ and

$$p_i \overline{u} - \sum_{j : i \prec j} \overline{z}_{ij} = 0 \quad i \in X \tag{12}$$

$$\sum_{i : i \prec j} \overline{z}_{ij} = w_j \quad j \in Y \tag{13}$$

$$\overline{u}, \overline{z}_{ij} \geq 0 \tag{14}$$

Notice, however, that Equations (12), (13), (14) define a transportation (minimum-cost network flow) problem on the precedence graph $G = (X \cup Y; E)$ with arc capacities $c(i, j) = \infty$ whenever $i \prec j$, and supply of $p_i \overline{u}$ at each $i \in X$ and demand of w_j at each $j \in Y$. The reader may notice some similarities with Lawler's maximum-flow construction [23] to compute a maximum-rank ideal of N. However, on closer inspection the two flow problems are quite different. We can choose

$\overline{u} = w(Y)/p(X)$ in order to make the total supply equal to the total demand. Furthermore, by Gale's theorem (see, e.g., [7]), this transportation problem has a feasible solution if and only if for any $S \subseteq Y$, $pred(S) = \{i \mid i \prec j\}$ has sufficient supply to satisfy the demand in S, i.e.,

$$\sum_{i \in pred(S)} p_i \overline{u} = (w(Y)/p(X))p(pred(S)) \geq w(S) \tag{15}$$

Condition (15), however, is equivalent to $w(S \cup pred(S))/p(S \cup pred(S)) \leq w(Y)/p(X)$ for every ideal $S \cup pred(S)$ in the precedence graph, i.e., the precedence graph must be indecomposable.

We also observe that for the complementary solutions considered, we have equality of the primal and dual objective function, i.e., $\sum_{j \in Y} w_j \overline{x}_j = w(Y)/l = (1/l)p(X)\overline{u} + \sum_{i \in X \cup Y} \overline{Z}_i = (1/l)p(X)\overline{u}$. Therefore, by the complementary slackness theorem of linear programming, $\overline{x}_i = 1/l$ for $i \in X \cup Y$ is an optimal solution of $LPOK$. ∎

Theorem 6.1 shows that when the input is indecomposable, the same fraction of weight and processing time will be assigned to the ideal by the optimal solution to $LPOK$, hence the algorithm from Lemma 5.2 cannot apply. Theorem 6.1 does not express only the limitations of our algorithm but the limitations of the relaxation as well. The linear program $LPOK$ can be seen as maximizing the rank of the ideal that meets the processing-time packing constraint. When the balanced solution in which all variables are equal is optimal, i.e., in the case of indecomposability, the rank computed is equal to $w(Y)/p(X)$, which is equivalent to selecting the entire set $X \cup Y$ as the ideal. Therefore no information is really obtained from $LPOK$ in this case. The next theorem gives an upper bound on the optimum of $LPOK$ for general inputs.

Theorem 6.2 *The fractional optimum of $LPOK$ is at most $(\lambda/l)w(Y)$ on an input with maximum-rank ideal of value $\lambda w(Y)/p(X)$, $\lambda \geq 1$.*

Proof. The case $\lambda = 1$ follows from Theorem 6.1. When the input is decomposable, let $r = \max\{w(S)/p(S) \mid S \text{ ideal }\}$. By decomposability, we know that $r = \lambda w(Y)/p(X)$, $\lambda > 1$. Setting $u = \lambda w(Y)/p(X)$, $Z_i = 0$ for all i and determining z_{ij} by a transportation problem gives a feasible solution to the dual program $DLPOK$ with objective value $p(X)u/l = \lambda w(Y)/l$. This allows us to upperbound the optimum value of the primal program $LPOK$. ∎

Finally we observe that the valid inequalities for Knapsack (without precedence constraints) which were recently proposed by Carr et al. [3] do not seem either to help in the difficult case, namely indecomposability. If we augment $LPOK$ with these inequalities, it is easy to see that the balanced solution is feasible for the new formulation as well.

7 Evidence of hardness

The positive result of Lemma 5.2 has two limitations: first it applies only to fractional solutions satisfying weight-majority and second it does not give an $O(1)$-approximation for the weight in the classical multiplicative sense. In this section we show that overcoming these limitations would lead to improved approximations for both the time-majority case and $1|prec|\sum w_j C_j$. The latter result would make progress on a longstanding open problem.

For the purposes of this section a POK instance is defined as (N, h, l) where h and l are scalars in $[1, +\infty)$. We seek an ideal N' that satisfies $p(N') \leq p(N)/l$ and $w(N') \geq w(N)/h$. (N, h, l) is a *weight-majority instance* if $h < l$, and a *time-majority instance* if $h > l$. It is easy to see that if a poset N is indecomposable then no weight-majority instance of POK on N can be feasible. A

ρ-approximation algorithm, $\rho < 1$, for a POK instance (N, h, l) yields, if the instance is feasible, an ideal of weight at least $\rho w(N)/h$ and processing time at most $p(N)/l$. If the instance is infeasible the algorithm should return the answer "NO". The idea behind the proof of the upcoming theorem is that even if a given set N has no ideal N' such that $w(N')/p(N') > w(N)/p(N)$, i.e., if N is indecomposable, we can add auxiliary weight so that a feasible weight-majority instance I' can be defined. If in turn we have an algorithm to extract from I' an ideal with large enough weight, this answer will contain a "good" fraction of the original weight from N.

Theorem 7.1 *Let δ be any constant in $(0, 1)$. If there is a δ-approximation algorithm $\mathcal{A}$ for weight-majority knapsack instances, then there exists an $O(1)$-approximation algorithm for a time-majority instance (N, h, l) where h and l are constants larger than 1 that depend on δ.*

Remark 7.1 *The lower the δ, the higher are h and l.*

Proof. Let (N, h, l) be a time-majority instance for which the range of h, l will be specified later on. We will reduce finding an $O(1)$-approximation for (N, h, l) to invoking $\mathcal{A}$ on an appropriate weight-majority instance. We create the latter instance (M, h', l) by setting $M = N \cup \{v\}$, where v is a new job with $p_v = 0$ and $w_v = Lw(N)$ for some $L > 0$. The precedence order on M is the same as the one on N with the addition of v as an independent job. The following claim is straightforward.

Claim 7.1 *Let $\xi \in (0, 1)$ be a constant. Any ideal M' of M of weight at least $(1+L-\xi)w(M)/(1+L)$ must contain $(1 - \xi)w(N)$ weight from jobs in $N \cap M$. Moreover the jobs in $N \cap M'$ form an ideal of N.*

We set $1/h' = (1 + L - c)/(1 + L)$ for some $c \in (0, 1)$. We then define h by the following equation

$$1 - c = 1/h. \tag{16}$$

We want (M, h', l) to be weight-majority and (N, h, l) to be time-majority instances, respectively. The condition for this is

$$1/h' = (1 + L - c)/(1 + L) > 1/l > 1 - c \tag{17}$$

For any $L, c > 0$ we have $(1 + L - c)/(1 + L) > 1 - c$. Therefore once these two parameters are fixed, we can choose an l to satisfy (17), assuming no other conditions on l are imposed. We also want a δ-approximation for (M, h', l) to imply an $O(1)$-approximation for (N, h, l). Suppose in particular we want a $(1 - c')/(1 - c)$-approximation for (N, h, l) for some $c' \in (c, 1)$. Since $(1/h)((1 - c')/(1 - c)) = 1 - c'$, by applying Claim 7.1 with $\xi = c'$, it is enough to require

$$\delta(1 + L - c)/(1 + L) \geq (1 + L - c')/(1 + L) \tag{18}$$

Constant δ is given from the hypothesis of the theorem. We now work backwards to ensure we can choose h', l, L, h to satisfy the conditions posed so far: (16), (17), (18), $0 < c < c' < 1$, and $L > 0$.

Inequality (17) can always be satisfied by choosing $1/l$ appropriately after fixing L, c. The same holds for Equation (16). Inequality (18) is equivalent to $\delta > (1 + L - c')/(1 + L - c)$ which in turn is equivalent to $L < (\delta(1 - c) - (1 - c'))/(1 - \delta)$. All we have to do is pick c and c' such that $\delta(1 - c) - (1 - c') > 0$. Then we can fix L accordingly. Fix c and c' such that

$$1 - c' < \delta(1 - c) \tag{19}$$

Inequality (19) is consistent with the requirement that $c' > c$.

We have shown that all the parameters can be chosen so that the statement of the theorem holds for any given δ. ∎

Theorem 7.1 shows that finding an $O(1)$-approximation for the weight-majority case is as hard as finding an $O(1)$-approximation for a class of time-majority instances with small but constant $1/h$ and $1/l$ parameters. The latter is a highly non-trivial problem as it has important consequences for the approximability of $1|prec|\sum w_jC_j$. Recall that if a set is indecomposable only time-majority instances can be feasible on that set. The $O(1)$-approximation of a time-majority POK instance on such an indecomposable set N would lead to a $2-\beta$ approximation for some fixed $\beta > 0$ for $1|prec|\sum w_jC_j$, however small $1/h$ and $1/l$. The latter statement can be shown easily using methods similar to the ones used in [32]. For the sake of completeness we provide a proof of the upcoming theorem in the Appendix, without attempting to maximize the constant β.

Theorem 7.2 *Let δ be any constant in $(0,1)$. If there is a δ-approximation algorithm $\mathcal{A}'$ for time-majority POK instances (N,h,l) where h and l are arbitrary constants larger than 1, then there is a $2-\beta$ approximation for some fixed $\beta > 0$ for $1|prec|\sum w_jC_j$.*

From Theorems 7.1 and 7.2 we obtain the following corollary.

Corollary 7.1 *Let δ be any constant in $(0,1)$. If there is a δ-approximation algorithm for weight-majority knapsack instances, there is a $2-\beta$ approximation for some fixed $\beta > 0$ for $1|prec|\sum w_jC_j$.*

Acknowledgment. Stavros Kolliopoulos thanks Gerhard Woeginger for useful discussions.

References

[1] Y. Asahiro, K. Iwama, H. Tamaki, and T. Tokuyama. Greedily finding a dense subgraph. *Journal of Algorithms*, 34:203–221, 2000.

[2] A. Brandstädt, V. B. Le, and J. P. Spinrad. *Graph Classes: A Survey.* SIAM, Philadelphia, PA, 1999.

[3] R. D. Carr, L. K. Fleischer, V. J. Leung, and C. A. Phillips. Strengthening integrality gaps for capacitated network design and covering problems. In *Proceedings of the 11th ACM-SIAM Symposium on Discrete Algorithms*, 2000.

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

[5] 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.

[6] F. A. Chudak and D. B. Shmoys. Approximation algorithms for precedence-constrained scheduling problems on parallel machines that run at different speeds. In *Proceedings of the 8th ACM-SIAM Symposium on Discrete Algorithms*, pages 581–590, 1997.

[7] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver. *Combinatorial Optimization.* John Wiley and Sons, New York, 1998.

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

[9] B. Dushnik and E.W. Miller. Partially ordered sets. *Amer. J. Math.*, 63:600–610, 1941.

[10] U. Feige. Relations between average case complexity and approximation complexity. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pages 534–543, 2002.

[11] U. Feige and S. Kogan. Hardness of approximation of the balanced complete bipartite subgraph problem. Technical Report MCS04-04, Department of Computer Science and Applied Math, The Weizmann Institute of Science, 2004.

[12] U. Feige, G. Kortsarz, and D. Peleg. The dense k-subgraph problem. *Algorithmica*, 29:410–421, 2001.

[13] M. X. Goemans and D. P. Williamson. Two-dimensional Gantt charts and a scheduling algorithm of Lawler. *SIAM Journal on Discrete Mathematics*, 13:281–294, 2000. Preliminary version in Proc. SODA 1999.

[14] M.T. Hajiaghayi, K. Jain, K. Konwar, L.C. Lau, I. I. Măndoiu, A. Russel, A. Shvartsman, and V.V. Vazirani. The minimum k-colored subgraph problem in haplotyping and DNA primer selection. To appear in Proc. Int. Workshop on Bioinformatics Research and Applications (IWBRA), 2006.

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

[16] D. S. Hochbaum. Approximating covering and packing problems: set cover, vertex cover, independent set, and related problems. In D. S. Hochbaum, editor, *Approximation Algorithms for NP-hard problems*, pages 94–143. PWS, 1997.

[17] O. H. Ibarra and C. E. Kim. Fast approximation algorithms for the Knapsack and sum of subset problems. *Journal of the ACM*, 22:463–468, 1975.

[18] D. S. Johnson and K. A. Niemi. On knapsacks, partitions, and a new dynamic programming technique for trees. *Mathematics of Operations Research*, 8(1):1–14, 1983.

[19] S. Khot. Ruling out PTAS for graph min-bisection, densest subgraph and bipartite clique. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, pages 136–145, 2004.

[20] T. Kloks and D. Kratsch. Computing a perfect edge without vertex elimination ordering of a chordal bipartite graph. *Information Processing Letters*, 55:11–16, 1995.

[21] T. Kloks and D. Kratsch. Treewidth of chordal bipartite graphs. *J. of Algorithms*, 19:266–281, 1995.

[22] S. G. Kolliopoulos and G. Steiner. Partially-ordered knapsack and applications to scheduling. In R. Möhring and R. Raman, editors, *Proceedings of the 10th Annual European Symposium on Algorithms*, volume 2461 of *Lecture Notes in Computer Science*, pages 612–624. Springer-Verlag, September 2002.

[23] E. L. Lawler. *Combinatorial Optimization: Networks and Matroids.* Holt, Rinehart and Winston, 1976.

[24] J. H. Lin and J. S. Vitter. ϵ-approximations with minimum packing constraint violation. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 771–782, 1992.

[25] F. Margot, M. Queyranne, and Y. Wang. Decompositions, network flows and a precedence constrained single-machine scheduling problem. *Operations Research*, 51(2000-29):981–992, 2003.

[26] R. H. Möhring. Computationally tractable classes of ordered sets. In I. Rival, editor, *Algorithms and Order*, pages 105–194. Kluwer, Boston, MA, 1989.

[27] R. Paige and R. E. Tarjan. Three partition refinement algorithms. *SIAM J. Computing*, 6:973–989, 1987.

[28] 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*, 131:655–663, 2003.

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

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

[31] W. T. Trotter. *Combinatorics and Partially Ordered Sets.* Johns Hopkins University Press, Baltimore, Md., 1992.

[32] G. J. Woeginger. On the approximability of average completion time scheduling under precedence constraints. In *Proceedings of the 28th ICALP*, pages 887–897, 2001.

[33] M. Yannakakis. On the complexity of partial order dimension problem. *SIAM J. Alg. Discr. Methods*, 3:351–358, 1982.

Appendix

The construct of 2-D Gantt charts [13] provides a nice geometric description of the proof of Theorem 7.2. A 2-D Gantt chart is defined in an *enclosing rectangle* in the positive orthant that has two corners at points $(0, 0)$ and $(p(N), w(N))$. A job j is represented in the chart by a rectangle R_j of width p_j and height w_j. A sequence $i_1, i_2, \ldots, i_n$ of jobs is represented by placing R_{i_1} with its lower left corner at $(0, 0)$ and R_{i_k} with its lower left corner coinciding with the upper right corner of $R_{i_{k-1}}$, for all $k = 2, \ldots, n$. The *covered area* of the sequence is equal to the area of the job rectangles plus the area that lies above those rectangles in the chart. It is easy to see that the total weighted completion time of a sequence is equal to its covered area. Let the main diagonal of the chart be the segment connecting $(0, 0)$ to $(p(N), w(N))$.

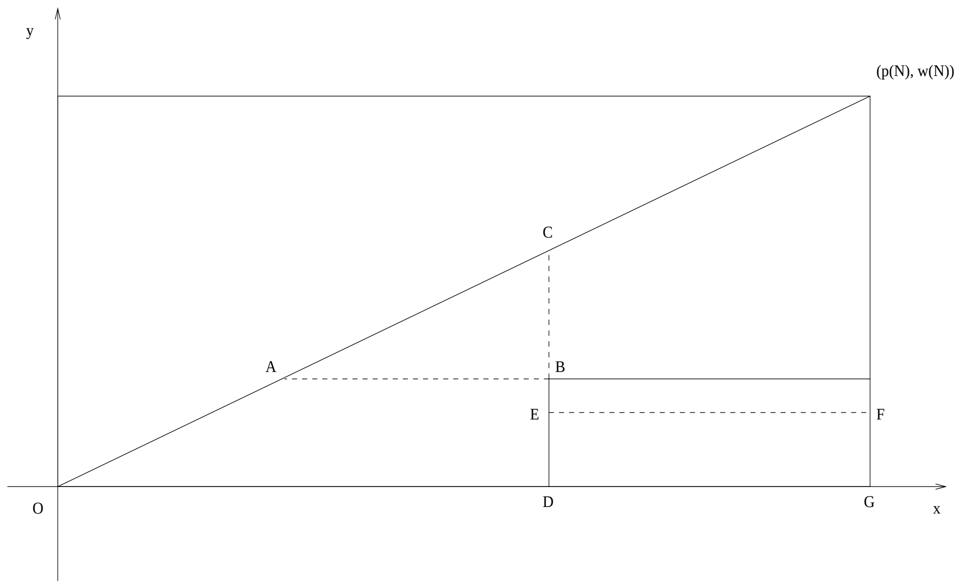

Figure 2: 2-D Gantt chart information used in the proof of Theorem 7.2. The coordinates of the points are: for D $(p(N)/l, 0)$, for B $(p(N)/l, w(N)/h)$, for E $(p(N)/l, \delta w(N)/h)$. A is the point on the diagonal with the same y-coordinate as B and C the point on the diagonal with the same x-coordinate as B.

Proof of Theorem 7.2. Let $P = (N, \prec_P)$ be the poset given as input to the scheduling problem. Based on the results of [30, 4] it suffices to focus our attention on indecomposable instances. Let (N, h, l) be a time-majority POK instance defined for *any* fixed $h, l > 1$ with $h > l$. Let λ be the rank of N, i.e., $\lambda = w(N)/p(N)$. We will design an approximation algorithm $\mathcal{B}$ for the scheduling problem.

Because of indecomposability, for any feasible sequence all the area above the main diagonal is covered [13]. Run the algorithm $\mathcal{A}'$ on the POK instance defined above and consider two cases.

Case 1. Algorithm $\mathcal{A}'$ returns "NO". Then algorithm $\mathcal{B}$ returns any feasible sequence that respects the precedence constraints as output.

Claim 7.2 *The triangle ABC defined in Fig. 2 belongs to the covered area of the optimal sequence S_o.*

Proof of claim. Let the optimal sequence S_o be $i_1, i_2, \dots, i_n$ and let j be the maximum index such that $\sum_{k=1}^{j} p_{i_k} \leq p(N)/l$. The upper right corners of the rectangles corresponding to the jobs $i_1, i_2, \dots, i_{j-1}$ lie in the interior of the trapezoid $ABDO$. The upper right corner of the rectangle corresponding to i_j lies in the interior of the same trapezoid or strictly below B on the segment DB. The lower right corner of the rectangle corresponding to job i_{j+1} lies strictly to the right of and below point B. ▮

The area of ABC is $(1/2)(BC)(AB)$. From the figure we obtain

$$(DC) = \lambda(OD) = \lambda p(N)/l = w(N)/l \quad \text{and}$$
$$(BC) = (DC) - (BD) = w(N)/l - w(N)/h = (h-l)w(N)/(hl) = cw(N),$$

where $c = (h-l)/(hl)$ and thus $0 < c < 1$. On the other hand $(AB) = (BC)/\lambda = cw(N)/\lambda = cp(N)$. Therefore $(1/2)(BC)(AB) = (c^2/2)p(N)w(N)$ and the total covered area of the optimal sequence S_o is at least

$$\frac{1+c^2}{2}p(N)w(N).$$

The covered area of the sequence produced by algorithm $\mathcal{B}$ is at most $p(N)w(N)$. Hence the approximation ratio achieved for the scheduling problem is at most $2/(1+c^2)$.

Case 2. Algorithm $\mathcal{A}'$ returns an ideal I of weight at least $\delta w(N)/h$ and processing time at most $p(N)/l$. Then algorithm $\mathcal{B}$ returns a sequence S that has the jobs of I first in any feasible order, followed by the jobs in $N \setminus I$ in any feasible order. The covered area of sequence S excludes the rectangle DEFG in Fig. 2. The area of DEFG is

$$(ED)(DG) \geq \delta\frac{w(N)(l-1)p(N)}{hl} = c'p(N)w(N),$$

where $c' = \delta(l-1)/(hl)$ and hence $0 < c' < 1$. Therefore the covered area of S is at most $(1-c')p(N)w(N)$. The approximation ratio achieved for the scheduling problem is at most $2(1-c')$.
$\blacksquare$