Approximating Single Machine Scheduling with Scenarios

Monaldo Mastrolilli, Nikolaus Mutsanas, and Ola Svensson

IDSIA - Lugano, Switzerland
{monaldo,nikolaus,ola}@idsia.ch

Abstract. In the field of robust optimization, the goal is to provide solutions to combinatorial problems that hedge against variations of the numerical parameters. This constitutes an effort to design algorithms that are applicable in the presence of uncertainty in the definition of the instance. We study the single machine scheduling problem with the objective to minimize the weighted sum of completion times. We model uncertainty by replacing the vector of numerical values in the description of the instance by a set of possible vectors, called *scenarios*. The goal is to find the schedule with minimum value in the worst-case scenario.

We first show that the general problem is intractable by proving that it cannot be approximated within $O(\log^{1-\varepsilon} n)$ for any $\varepsilon > 0$, unless NP has quasi-polynomial algorithms. We then study more tractable special cases and obtain an LP based 2-approximation algorithm for the unweighted case. We show that our analysis is tight by providing a matching lower bound on the integrality gap of the LP. Moreover, we prove that the unweighted version is NP-hard to approximate within a factor less than 6/5. We conclude by presenting a polynomial time algorithm based on dynamic programming for the case when the number of scenarios and the values of the instance are bounded by some constant.

1 Introduction

In classical optimization problems, it is often assumed that the parameters of the instances are precisely defined numerical values. In many cases, however, such a precise definition is impossible due to inadequate knowledge on the side of the decision maker. The necessity to provide algorithms for minimizing the cost in uncertain environments lead to the fields of stochastic and robust optimization.

In *stochastic optimization* [4], it is assumed that we have knowledge of the probability distribution of the data and the goal is to find a solution that minimizes the expected cost. *Robust optimization* [3,15] can be considered as the worst-case counterpart of the stochastic optimization. In a robust optimization problem, we have a set of possible configurations of the numerical parameters of the problem, and the goal is to find a solution that minimizes the cost in a worst-case scenario for the given solution. In the following we will focus on this latter approach.

Within robust optimization, two common ways of modeling uncertainty are *interval data* and *discrete scenarios*. In the case of interval data the vector of

A. Goel et al. (Eds.): APPROX and RANDOM 2008, LNCS 5171, pp. 153–164, 2008.

numerical parameters in the description of the instance is replaced by a vector of intervals, one for each parameter. On the other hand, in the case of discrete scenarios the vector of numerical parameters is replaced by a set of vectors, each of them corresponding to a different scenario. An advantage of this model is that, whereas in the case of interval data the fluctuations of the different numerical parameters are implicitly assumed to be independent, the use of discrete scenarios allows the implementation of dependencies among parameters.

Several objective functions for robust minimization[1] problems have been proposed in literature (see e.g. the book by Kouvelis & Yu [15]). In the *absolute robustness* approach, the goal is to minimize the maximum among all feasible solutions and all scenarios. This is often referred to as the "min-max" version of the problem. In the *robust deviation* approach, the goal is to minimize the maximum deviation from optimality among all feasible solutions and all scenarios. Recent examples of these two families of approaches can be found in [1,12,9].

In this paper we investigate the min-max version of the following classical scheduling problem. There is a set $N = \{1, \ldots, n\}$ of n jobs to be scheduled 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. Jobs must be processed for p_j time units without interruptions on the machine. The goal is to find a schedule (i.e. permutation $\pi : N \to \{1, \ldots, n\}$) such that 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, is minimized. In standard scheduling notation (see e.g. Graham et al. [10]), this problem is known as $1||\sum w_j C_j$. Smith [22] gave a simple polynomial time algorithm for this problem, by showing that scheduling jobs in non-decreasing order of the ratio of their processing time to their weight is optimal: given a set of n jobs with weights w_j and processing times p_j, $1 \leq j \leq n$, schedule the jobs such that $\pi(i) < \pi(j)$ if and only if $p_i/w_i \leq p_j/w_j$. When there are precedence constraints among jobs, then the problem becomes NP-hard [16]. Several 2-approximation algorithms are known for this variant [19,11,6,5,17], as observed in [7], all of them can be seen as obtained by rounding a linear relaxation of an integer program formulation ILP due to Potts [18]. The integrality gap of ILP is known [5] to be 2, and understanding if a better than 2-approximation algorithm exists is considered an outstanding open problem in scheduling theory (see e.g. [21]). In this paper we consider the robust version of this classical scheduling problem, as defined below.

Definition 1. *In the robust scheduling problem, we are given a set of jobs* $N = \{1, \ldots, n\}$ *and a set of scenarios* $S = \{s_1, \ldots, s_m\}$ *where* $s_i = (p_1^{s_i}, \ldots, p_n^{s_i}, w_1^{s_i}, \ldots, w_n^{s_i})$ *for* $s_i \in S$*. A feasible schedule is a permutation* π *of the jobs and the problem is to find a permutation* π^* *of the jobs such that*

$$\pi^* = \min_{\pi} \max_{s_i \in S} \left(\sum_{j \in N} w_j^{s_i} C_j^{s_i}(\pi) \right),$$

where $C_j^{s_i}(\pi) = \sum_{j' \in N, \pi(j') \leq \pi(j)} p_{j'}^{s_i}$.

[1] The definition for robust maximization problems are analogous.

Whereas $1||\sum w_jC_j$ is polynomial time solvable in the case of a single scenario, Kouvelis & Yu [15] prove that the robust version is weakly NP-complete even for the case of two scenarios and unit processing times.

In this paper we take on the task of studying the approximability of the robust variant. We show that, unless NP has quasi-polynomial algorithms, it cannot be approximated within factor $O(\log^{1-\varepsilon} n)$ in polynomial time, for any $\varepsilon > 0$. Moreover, under $P \neq NP$, we show that it remains hard to approximate within $6/5$ even if we assume that processing times, or alternatively weights, are equal to one and do not vary across the scenarios.

Then, we study the natural generalization of the ILP due to Potts [18] for the robust version. We provide a lower bound on the integrality gap and a matching upper bound for the special case where processing times or, alternatively, weights do not vary across the scenarios. Interestingly, the upper bound can be extended to include precedence constraints, and we obtain the same performance guarantee, namely a 2-approximation, as for the single scenario case. Proving good hardness of approximation results for $1|prec|\sum w_jC_j$ is a long standing open problem in scheduling theory. In contrast, for the robust variant, we show that it is NP-hard to approximate within a factor less than $6/5$.

We conclude by presenting a polynomial time algorithm based on dynamic programming for the case that the number of scenarios and the values of the instance are bounded by some constant.

2 Hardness of the Robust Scheduling Problem

2.1 Inapproximability Result for the General Problem

Here, we show that the general problem with non-constant number of scenarios has no $O(\log^{1-\varepsilon} n)$-approximation algorithm for any $\varepsilon > 0$, unless NP has quasi-polynomial algorithms. The hardness result is obtained by reducing the following version of the Label Cover problem to the scheduling problem.

Definition 2. *The Label Cover problem* $\mathcal{L}(V, W, E, [R], \{\sigma_{v,w}\}_{(v,w)\in E})$ *is defined as follows. We are given a regular bipartite graph with left side vertices* V*, right side vertices* W*, and set of edges* $E \subseteq V \times W$*. In addition, for every edge* $(v, w) \in E$ *we are given a map* $\sigma_{v,w} : [R] \to [R]$*. A labeling of the instance is a function* ℓ *assigning a set of labels to each vertex of the graph, namely* $\ell : V \cup W \to 2^{[R]}$*. A labeling* ℓ *satisfies an edge* (v, w) *if*

$$\exists a \in \ell(v), \exists b \in \ell(w) : \sigma_{v,w}(a) = b.$$

A total-labeling is a labeling that satisfies all edges. The value of a Label Cover instance, denoted $val(\mathcal{L})$*, is defined to be the minimum, over all total-labelings, of* $\max_{x\in V\cup W} |\ell(x)|$.

Observe that the variant of the Label Cover problem that is considered assumes that an edge is covered if, among the chosen labels, there *exists* a satisfying pair of labels. The following hardness result easily follows from the hardness result

for the max version by using the "weak duality" relationship between the two versions (see e.g. [2]).

Theorem 1. *There exists a constant $\gamma > 0$ so that for any language L in NP, any input w and any $R > 0$, one can construct a labeling instance $\mathcal{L}$, with $|w|^{O(\log R)}$ vertices, and label set of size R, so that: If $w \in L, val(\mathcal{L}) = 1$ and otherwise $val(\mathcal{L}) > R^{\gamma}$. Furthermore, $\mathcal{L}$ can be constructed in time polynomial in its size.*

We prove the following theorem by presenting a reduction from the label cover problem.

Theorem 2. *There exists a constant $\gamma > 0$ so that for any language L in NP, any input w, any $R > 0$ and for $g \leq R^{\gamma}$, one can, in time $O(|w|^{O(g \log R)} \cdot R^{O(g)})$, construct a robust scheduling instance that has optimal value $1 + o(1)$ if $w \in L$ and optimal value g otherwise.*

Proof. Given a Label Cover instance $\mathcal{L}(V, W, E, [R], \{\sigma_{v,w}\}_{(v,w)\in E})$, we construct a robust scheduling instance I. Before giving a more formal definition of the reduction, we first give the intuition behind it.

For $x \in V \cup W$ let $R_x \subseteq [R]$ be the possible labels of x. For each $(v, w) \in E$, let $R_{v,w} \subseteq R_v \times R_w$ contain all pairs of labels of v and w that satisfy the map $\sigma_{v,w}$, i.e., $R_{v,w} = \{(a, b) \in R_v \times R_w : b = \sigma_{v,w}(a)\}$.

Clearly, for any feasible label cover ℓ there is at least one pair (a, b) from $R_{v,w}$ such that $a \in \ell(v)$ and $b \in \ell(w)$, and we say that (a, b) *covers* (v, w). In order to have a "corresponding" situation in the scheduling instance I, we define for each $(v, w) \in E$ a set $\mathcal{J}^{(v,w)} = \{J_1^{(v,w)}, J_2^{(v,w)}, \ldots, J_{n_{v,w}}^{(v,w)}\}$ of $n_{v,w} = |R_{v,w}|$ jobs. Let us consider some total ordering $r_{v,w} : R_{v,w} \to \{1, \ldots, n_{v,w}\}$ of the pairs in $R_{v,w}$. In any feasible schedule of the jobs from $\mathcal{J}^{(v,w)}$ there exists an $i = 1, \ldots, n_{v,w}$, such that $J_{i+1}^{(v,w)}$ is scheduled before $J_i^{(v,w)}$ (assume $i + 1$ equal to 1 when $i = n_{v,w}$), otherwise we would have a cycle in that schedule. The reduction that we are going to present will associate this situation (job $J_{i+1}^{(v,w)}$ scheduled before $J_i^{(v,w)}$) to the case where the i^{th} pair in $R_{v,w}$ is in the label cover, i.e. $r_{v,w}^{-1}(i)$ covers edge (v, w). Then, for each $x \in V \cup W$, a set of scenarios is defined such that the maximum value of them counts (up to g) the number of different labels of x. A precise description of the reduction is given below.

Jobs. The jobs of instance I are the union of all jobs $\bigcup_{(v,w)\in E} \mathcal{J}^{(v,w)}$.

Ordering Scenarios. Let $m = |E|$ and let $\pi : E \to \{1, \ldots, m\}$ be some order of the edges. For each $i : 1 \leq i < m$, we have a scenario that sets the weights of the jobs in $\mathcal{J}^{\pi^{-1}(i)}$ to m and the processing time of the jobs in $\bigcup_{j>i} \mathcal{J}^{\pi^{-1}(j)}$ to m. The purpose of these scenarios is to ensure that any optimal schedule will schedule the jobs in the order

$$\mathcal{J}^{\pi^{-1}(1)} \prec \mathcal{J}^{\pi^{-1}(2)} \prec \cdots \prec \mathcal{J}^{\pi^{-1}(m)}. \tag{1}$$

Counting Scenarios. For each $v \in V$, let $E_v \subseteq E$ denote the set of edges incident to v. For each tuple $((v,w_1),\ldots,(v,w_g)) \in E_v \times \cdots \times E_v$ of pairwise different edges, for each tuple $(a_1,\ldots,a_g) \in R_v \times \cdots \times R_v$ of pairwise different labels, and for each tuple $(b_1,\ldots,b_g) \in R_{w_1} \times \cdots \times R_{w_g}$ so that $\sigma_{(v,w_i)}(a_i) = b_i$ for $i = 1,\ldots,g$, we have a different scenario $\mathcal{S}^{(v,w_1),\ldots,(v,w_g)}_{(a_1,b_1),\ldots,(a_g,b_g)}$. Each scenario $\mathcal{S}^{(v,w_1),\ldots,(v,w_g)}_{(a_1,b_1),\ldots,(a_g,b_g)}$ represents the situation in which label (a_i,b_i) covers edge (v,w_i) and the number of different labels of v is at least g. This label cover (partial) solution corresponds to the scheduling solutions $\sigma^{(v,w_1),\ldots,(v,w_g)}_{(a_1,b_1),\ldots,(a_g,b_g)}$ that schedule job $J^{(v,w_i)}_{h+1}$ before $J^{(v,w_i)}_{h}$, where $h = r_{v,w_i}(a_i,b_i)$, for each $i = 1,\ldots,g$. The value of these schedules is made larger than g by setting the processing time of $J^{(v,w_i)}_{h+1}$ equal to $m^{2\pi(v,w_i)}$ and the weight of $J^{(v,w_i)}_{h}$ equal to $1/m^{2\pi(v,w_i)}$, for each $i = 1,\ldots,g$, and zero all the others. Observe that the processing times and weights have been picked in such a way that jobs in $\mathcal{J}^{\pi^{-1}(i)}$ only contribute a negligible amount to the weighted completion time of jobs in $\mathcal{J}^{\pi^{-1}(j)}$ for $i < j$. This defines weights and processing times of scenarios for every $v \in V$. In a symmetric way we define scenarios $\mathcal{S}^{(v_1,w),\ldots,(v_g,w)}_{(a_1,b_1),\ldots,(a_g,b_g)}$, for every $w \in W$, to count the number of labels that are assigned to w.

The total number of scenarios is at most $|E| - 1 + 2|E|^g \cdot R^g \cdot R^g$ and the total number of jobs is at most $|E| \cdot R^2$. As $|E| = |w|^{O(\log R)}$, the total size of the robust scheduling instance is $O(|w|^{O(g \log R)} \cdot R^{O(g)})$.

Completeness Analysis. By Theorem 1, there exists a feasible labeling of $\mathcal{L}$ that assigns one label to each vertex. Let ℓ be such a labeling and consider a schedule σ of I that respects (1) and such that, for each element $(v,w) \in E$, the jobs in $\mathcal{J}^{(v,w)}$ are scheduled as follows: for $h = 1,\ldots,n_{v,w}$, if $h = r_{v,w}(\ell(v),\ell(w))$ then job $J^{(v,w)}_{h+1}$ is scheduled before $J^{(v,w)}_{h}$, otherwise $J^{(v,w)}_{h}$ is before $J^{(v,w)}_{h+1}$. This gives a feasible schedule. Moreover, since only one label is assigned to each vertex, it is easy to see that the value of any scenario is at most $1 + o(1)$.

Soundness Analysis. Consider a schedule σ of I. Define a labeling ℓ as follows:

$$\begin{cases} \ell(v) = \{a : \text{if } J^{(v,w)}_{h+1} \prec J^{(v,w)}_{h} \text{ for some } h = r_{v,w}(a,b), w \in W \text{ and } b \in R_w\} \\ \ell(w) = \{b : \text{if } J^{(v,w)}_{h+1} \prec J^{(v,w)}_{h} \text{ for some } h = r_{v,w}(a,b), v \in V \text{ and } a \in R_v\} \end{cases}$$

As at least one scenario for each edge will have value 1, ℓ is a feasible labeling of $\mathcal{L}$. Furthermore, by Theorem 1, there exists a vertex $x \in V \cup W$ so that $|\ell(x)| \geq g$, and this implies that there is a scenario of value g. Indeed, if $x \in V$ let $\ell(x) = \{a_1,\ldots,a_g\}$ be the set of g labels assigned to x, and let $(b_1,\ldots,b_g)$ and $(w_1,\ldots,w_g)$ be such that $J^{(v,w)}_{h+1} \prec J^{(x,w)}_{h}$ with $h = r_{x,w_i}(a_i,b_i)$. Then scenario $\mathcal{S}^{(x,w_1),\ldots,(x,w_g)}_{(a_1,b_1),\ldots,(a_g,b_g)}$ has been constructed to have value g according to this schedule. The same holds when $x \in W$.

By setting $g = O(\log^c n)$ (and $R = O(\log^{O(c)} n)$), where $|w| = n$ and $c \geq 1$ any large constant, we obtain that the input size is equal to $s = n^{O(g \log R)} \cdot R^{O(g)} = n^{O(\log^c n \cdot \log\log n)} \cdot (\log n)^{O(\log^c n)} = n^{O(\log^{c+\delta} n)} = 2^{O(\log^{c+1+\delta} n)}$, for any arbitrarily small $\delta > 0$. It follows that $g = O(\log s)^{\frac{c}{c+1+\delta}} = O(\log s)^{1-\varepsilon}$, for any arbitrarily small $\varepsilon > 0$.

Theorem 3. *For every $\varepsilon > 0$, the robust scheduling problem cannot be approximated within ratio $O(\log^{1-\varepsilon} s)$, where s is the input size, unless NP has quasi-polynomial algorithms.*

2.2 Inapproximability for Unit-Time/Unweighted Case

We now restrict the above problem to the case where the processing times do not vary across scenarios. We note that this case is symmetric to the one where the processing times may vary across scenarios while the weights are common. We show that, if the number of scenarios is unbounded, the robust scheduling problem is not approximable within 6/5 even for the special case that all processing times are equal to one.

Our reduction is from the E3-Vertex-Cover problem, defined as follows. Given a 3-uniform hypergraph $G = (V, E)$ (each edge has size 3), the E3-Vertex-Cover problem is to find a subset $S \subseteq V$ that "hits" every edge in G, i.e. such that for all $e \in E, e \cap S \neq \emptyset$. Dinur et al. [8] showed that it is NP-hard to distinguish whether a k-uniform hypergraph has a vertex cover of weight $(\frac{1}{k-1} + \varepsilon)n$ from those whose minimum vertex cover has weight at least $(1 - \varepsilon)n$ for an arbitrarily small $\varepsilon > 0$.

Given a 3-uniform hypergraph $G(V, E)$, we construct a robust scheduling instance as follows.

- For every vertex $v_i \in V$ we create a job $i \in N$ with $p_i = 1$.
- For every hyperedge $e = \{v_1^e, v_2^e, v_3^e\} \in E$ we create a scenario s_e defined by

$$w_i^{s_e} = \begin{cases} 1\ , & \text{if } v_i \in \{v_1^e, v_2^e, v_3^e\} \\ 0\ , & \text{otherwise.} \end{cases}$$

Given the size of a minimum vertex cover c, one can calculate upper and lower bounds on the optimal value of the corresponding scheduling instance, as follows: given a schedule, i.e., a permutation π of the jobs, we can define a vertex cover solution VC by letting

$$VC = \{v_i \mid v_i \text{ covers an edge not covered by } \{v_j | \pi(j) < \pi(i)\}\}.$$

Let $v_j \in VC$ be the vertex in VC that is scheduled last, i.e., any $v_i \in VC$ with $i \neq j$ satisfies $\pi(i) < \pi(j)$. As v_j was added to VC, it covers an edge, say $e = \{v_j, v_k, v_l\}$, with $\pi(j) < \pi(k)$ and $\pi(j) < \pi(l)$. Furthermore, since $|VC| \geq c$ we have that $\pi(j) \geq c$ and hence $\pi(k) + \pi(l) \geq (c + 1) + (c + 2)$. It follows that there is an $s \in S$ with value at least

$$LB(c) = c + (c + 1) + (c + 2) > 3c$$

which is thus also a lower bound on $\min_{\pi} \max_{s \in S} (\text{val}(\pi, s))$.

For the upper bound, consider the schedule where we schedule c jobs corresponding to a minimum vertex cover first. Observe that a scenario in which the last of these c jobs has weight one takes its maximal value if the other two jobs of the corresponding edge are scheduled last, yielding

$$UB(c) = c + (n-1) + n < c + 2 \cdot n.$$

Using the inapproximability results of Dinur et al. [8] we get the following gap for the robust scheduling problem:

$$\frac{LB((1-\varepsilon)n)}{UB((\frac{1}{3-1}+\varepsilon)n)} > \frac{(1-\varepsilon)n \cdot 3}{(\frac{1}{2}+\varepsilon)n + 2 \cdot n} = \frac{6}{5} - \varepsilon'$$

for some $\varepsilon' > 0$ that can be made arbitrarily small. As the unit-time and unweighted robust scheduling problem are symmetric, this yields the following theorem.

Theorem 4. *It is NP-hard to approximate the unit-time/unweighted robust scheduling problem within a factor less than* $6/5$.

Assuming the *Unique Games Conjecture* [13], the inapproximability result for Ek-uniform Vertex Cover improves to a gap of $k - \varepsilon$ [14]. A similar reduction from 2-uniform hypergraphs (i.e. graphs) using the same bounds as above yields an inapproximability gap of $4/3$.

Finally, we note that an easy numerical analysis shows that, in both cases, the inapproximability results cannot be improved by changing the uniformity of the hypergraphs in the vertex cover problems considered.

3 An LP-Based Approximation Algorithm and Integrality Gap

In this section, we consider the special case that processing times do not vary among scenarios, i.e. for every $i \in N$ we have $p_i^{s_1} = \ldots = p_i^{s_m} = p_i$. Note that this is symmetric to the case that processing times may vary across scenarios while weights are common. Inspired by Potts [18] integer linear program (ILP) formulation of $1|prec|\sum w_j C_j$, we formulate the robust scheduling problem with common processing times as follows:

$$\begin{aligned}
\min \quad & t & & \\
\sum_{j \in N} p_j w_j^{s_k} + \sum_{(i,j) \in N^2} \delta_{ij} \cdot p_i w_j^{s_k} &\leq t & & 1 \leq k \leq m \\
\delta_{ij} + \delta_{ji} &= 1 & & (i,j) \in N^2 \\
\delta_{ij} + \delta_{jk} + \delta_{ki} &\geq 1 & & (i,j,k) \in N^3 \\
\delta_{ij} &\in \{0,1\} & & (i,j) \in N^2
\end{aligned}$$

The variables, δ_{ij} for $(i,j) \in N^2$, are called ordering variables with the natural meaning that job i is scheduled before job j if and only if $\delta_{ij} = 1$. The LP

relaxation of the above ILP is obtained by relaxing the constraint $\delta_{ij} \in \{0,1\}$ to $\delta_{ij} \geq 0$. We will show that the resulting LP has an integrality gap of 2.

Consider the following family of instances, consisting of n jobs and an equal number of scenarios. The (scenario-independent) processing times are set to $p_j = 1,\ j \in N$. The weights of the jobs in scenario s_k are defined as follows:

$$w_j^{s_k} = \begin{cases} 1\ , \text{ if } j = k \\ 0\ , \text{ otherwise} \end{cases},\ j \in N.$$

It is easy to see that setting

$$\delta_{ij} = 1/2, \quad 1 \leq i, j \leq n,\ i \neq j$$

yields a feasible solution. For this solution, all scenarios assume the same objective value

$$p_j + \sum_{i \neq j} \delta_{ij} p_i = 1 + (n-1) \cdot \frac{1}{2} = \frac{n+1}{2}$$

which therefore equals the objective value of this solution. This gives an upper bound on the value of the optimal solution.

On the other hand, for any feasible integral solution, there is a scenario s_k for which the job j is scheduled last. This scenario has value $w_j^k \cdot C_j = n$. Thus the integrality gap of the above presented LP with n scenarios is at least $2n/(n+1)$, which tends to 2 as n tends to infinity.

We now provide a 2-approximation algorithm based on the above LP-relaxation, thus showing that the analysis of the integrality gap is tight.

Given a solution of the LP, let

$$\tilde{C}_j = p_j + \sum_{i \neq j} \delta_{ij} p_i$$

be the fractional completion time of job j. Assume, without loss of generality, that $\tilde{C}_1 \leq \ldots \leq \tilde{C}_n$. We will use the following property to derive a 2-approximation algorithm.

Lemma 1 (Schulz [20]). *Given a solution of the above LP, with $\tilde{C}_1 \leq \ldots \leq \tilde{C}_n$ the following inequality holds*

$$\tilde{C}_j \geq \frac{1}{2} \sum_{i=1}^{j} p_i$$

This property can be used to derive a simple 2-approximation algorithm: schedule the jobs in non-decreasing order of $\tilde{C}_j$. The integral completion time is

$$C_j = \sum_{i=1}^{j} p_i \leq 2 \cdot \tilde{C}_j.$$

Since every completion time increases by at most a factor of 2, we have a 2-approximate solution.

It is worth noting that the above analysis holds also for the case that there are precedence constraints among the jobs, a significant generalization of this problem. For instance, in the single scenario case, $1|prec|\sum w_jC_j$ is NP-complete whereas $1||\sum w_jC_j$ is polynomial time solvable. We summarize with the following theorem.

Theorem 5. *The robust version of* $1|prec|\sum w_iC_i$ *has a polynomial time* 2*-approximation algorithm when the processing times or, alternatively, the weights of the jobs do not vary among the scenarios.*

4 A Polynomial Time Algorithm for Constant Number of Scenarios and Constant Values

In this section we assume that the number of scenarios m as well as the weights and processing times are bounded by some constant. Given an instance I of the robust scheduling problem, let W be the maximum weight and P the maximum processing time occurring in the description of I. We present a polynomial time algorithm that solves this problem. In fact, we are going to solve the related multi-criteria scheduling problem. This result carries over to our problem by use of Theorem 1 in Aissi et. al. [1].

In the context of multi-criteria optimization, given two vectors $v, w \in \mathbb{N}^k$, $v \neq w$, $k > 0$, we say that v *dominates* w, if $v_i \leq w_i$ for all $1 \leq i \leq k$. A vector that is not dominated is called *efficient*. Analogously, given a set of vectors S, a subset $S' \subseteq S$ is called an *efficient set* if there is no pair $(v, v'), v \in S, v' \in S'$ such that v dominates v'. The goal in multi-criteria optimization is to find a maximal efficient set of solutions.

For a fixed set of scenarios $S = \{s_1, \ldots, s_m\}$, we define the *multivalue* of a schedule π by $\text{val}(\pi) = (\text{val}(\pi, s_1), \ldots, \text{val}(\pi, s_m))$. Furthermore, we call $\alpha = ((w_1, p_1), \ldots, (w_m, p_m))$ with $1 \leq w_i \leq W$, $1 \leq p_i \leq W$ a *job profile*, and let $p(\alpha) = (p_1, \ldots, p_m)$ and similarly $w(\alpha) = (w_1, \ldots, w_m)$. Note that, since we assumed that P,W and m are all bounded by a constant, there can only be a constant number of different job profiles. Let $\alpha_1, \ldots, \alpha_k$ be the different job profiles that occur in instance I. We can now identify I by the tuple $((\alpha_1, \ldots, \alpha_k), (n_1, \ldots, n_k))$ where n_i is the number of jobs with profile α_i occurring in I. We will present a dynamic programming approach for solving the min-max scheduling problem with a constant number of scenarios and constant values in polynomial time.

4.1 Polynomial Time Algorithm

Consider a k-dimensional dynamic programming table DPT of size $(n_1 + 1) \times (n_2 + 1) \times \ldots \times (n_k + 1)$. Each cell of this table represents a subinstance I' of I, where the coordinates of the cell encode the number of jobs of the corresponding profile that are present in I' (for instance, the cell $(1, 0, 4)$ represents the subinstance of I that contains one job of type α_1 and four jobs of type α_3). We

denote the number of jobs in an instance represented by a cell $c = (c_1, \ldots, c_k)$ by $n(c) = \sum_{i=1}^{k} c_i$. Each of these cells will accommodate an efficient set M_c of multivalues of schedules in which only the jobs of the subinstance are considered (note that since the maximum value in any scenario is bounded, there can only be a polynomial number of different efficient vectors). Since the cell $(n_1, \ldots, n_k)$ represents the whole instance, filling in the last cell of the table would allow us to solve the multi-criteria scheduling problem, and thus also the min-max scheduling problem.

We initialize the table by filling in the cells whose coordinates sum up to one, i.e. the cells $c = (c_1, \ldots, c_k)$ with $n(c) = 1$, as follows: for $c_t = 1$ add to M_c the multivalue of the schedule consisting of a single job with profile α_t. We continue filling in the rest of the cells in order of increasing $n(c)$ in the following manner.

Consider the cell c with coordinates $(c_1, \ldots, c_k)$ and let $T_c = \{(c'_1, \ldots, c'_k) \mid n(c') = n(c) - 1,\ c'_i \geq c_i - 1\}$. In other words, T_c contains those cells representing subinstances that result by removing one job from I_c. Note that, since we fill in the table in order of increasing $n(c)$, all cells in T_c have been filled in at this point. For each $c' \in T_c$ with $c_t - c'_t = 1$, add to the set M_c the multivalues of the schedules that result from the schedules in $M_{c'}$ by adding a job of profile α_t in the end of the schedule. More formally, for each π' with $\mathrm{val}(\pi') \in M_{c'}$, add $\mathrm{val}(\pi)$ to M_c, with π defined as follows:

$$\pi(j) = \pi'(j) \text{ for } 1 \leq j \leq n(c') \text{ and } \pi(n(c)) = \alpha_t.$$

Given $\mathrm{val}(\pi')$, the multivalues of these schedules can easily be computed by:

$$\mathrm{val}(\pi) = \mathrm{val}(\pi') + w(\alpha_t) \cdot \sum_{i=1}^{k} c_i \cdot p(\alpha_i)$$

Note that only the multivalue of π' is needed in the above calculations, not π' itself.

We conclude the computation for cell c by replacing M_c by $Red(M_c)$, which retains only the efficient elements of M_c.

Lemma 2. *For every cell c of the table DPT, the set M_c is a maximal efficient subset of the set of all multivalues achieved by scheduling the jobs of I_c.*

Proof. We need to show that for every cell c of the table DPT and every multivalue $\mathrm{val}(\pi)$, where π is a schedule of I_c, either

- $\mathrm{val}(\pi) \in M_c$, or
- $\exists v \in M_c$, such that $v \leq \mathrm{val}(\pi)$

Suppose, towards contradiction, that this is not the case, and let c be a cell with minimal $n(c)$ that does not satisfy the above condition. Thus, there is a schedule π of the instance I_c with $\mathrm{val}(\pi) \notin M_c$ and for any $v \in M_c$ there is an $l \in \{1, \ldots, k\}$ with $\mathrm{val}(\pi)_l < v_l$. Clearly, this can only happen for $n(c) \geq 2$. Let α_f be the profile of the job scheduled last in π and let c' be the cell

with coordinates $(c_1, c_2, \ldots, c_{f-1}, c_f - 1, c_{f+1}, \ldots, c_k)$. Furthermore, let π' be the schedule derived from π by omitting the last job. The multivalue of π' is $\mathrm{val}(\pi) - w(\alpha_f) \cdot \sum_{i=1}^{k} c_i \cdot p(\alpha_i)$. If there were a $v \in M_c$ such that $\mathrm{val}(v) \leq \mathrm{val}(\pi')$, then $\mathrm{val}(\pi)$ would be dominated by $v + w(\alpha_f) \cdot \sum_{i=1}^{k} c_i \cdot p(\alpha_i)$. Thus, for every $v \in M_{c'}$, there is an $l \in \{1, \ldots, k\}$ such that $v_l > \mathrm{val}(\pi')_l$ and thus c' does not satisfy the above property either. Since $n(c') < n(c)$, this contradicts the minimality of c.

It is easy to see that the initialization of the table, as well as the computations of $\mathrm{val}(\pi)$ can be done in polynomial time. Furthermore, since $(n^2 \cdot P \cdot W)^2$ is an upper bound on the value of any schedule in any scenario, there can be at most $(n^2 \cdot P \cdot W)^{2m}$ efficient vectors in any stage of the computation. The size of the dynamic programming table is bounded by n^k and for each computation of a cell, at most k cells need to be considered. Moreover, the operator *Red* can be implemented in time $(n^2 \cdot P \cdot W)^{4m}$ by exhaustive comparison. Thus, a single cell can be filled-in in time $k(n^2 \cdot P \cdot W)^{2m} + (n^2 \cdot P \cdot W)^{4m}$, and the whole table in time $n^k \cdot (k \cdot (n^2 \cdot P \cdot W)^{2m} + (n^2 \cdot P \cdot W)^{4m})$. The number of different profiles k is bounded by $(P \cdot W)^m$, which is a constant. Thus our algorithm runs in time $O(n^{8m+W^m P^m})$, i.e. polynomial in n.

Acknowledgements

This research is supported by Swiss National Science Foundation project 200021-104017/1, "Power Aware Computing", by the Swiss National Science Foundation project 200020-109854, "Approximation Algorithms for Machine scheduling Through Theory and Experiments II", and by the Swiss National Science Foundation project PBTI2-120966, "Scheduling with Precedence Constraints".

References

1. Aissi, H., Bazgan, C., Vanderpooten, D.: Approximating min-max (regret) versions of some polynomial problems. In: Chen, D.Z., Lee, D.T. (eds.) COCOON 2006. LNCS, vol. 4112, pp. 428–438. Springer, Heidelberg (2006)
2. Arora, S., Lund, C.: Hardness of approximations. In: Hochbaum, D.S. (ed.) Approximation Algorithms for NP-Hard Problems. PWS (1995)
3. Bertsimas, D., Sim, M.: Robust discrete optimization and network flows. Programming Series B 98, 49–71 (2002)
4. Birge, J., Louveaux, F.: Introduction to stochastic programming. Springer, Heidelberg (1997)
5. Chekuri, C., Motwani, R.: Precedence constrained scheduling to minimize sum of weighted completion times on a single machine. Discrete Applied Mathematics 98(1-2), 29–38 (1999)
6. Chudak, F.A., Hochbaum, D.S.: A half-integral linear programming relaxation for scheduling precedence-constrained jobs on a single machine. Operations Research Letters 25, 199–204 (1999)

7. Correa, J.R., Schulz, A.S.: Single machine scheduling with precedence constraints. Mathematics of Operations Research 30(4), 1005–1021 (2005); Extended abstract in Proceedings of the 10th Conference on Integer Programming and Combinatorial Optimization (IPCO 2004), pp. 283–297 (2004)
8. Dinur, I., Guruswami, V., Khot, S., Regev, O.: A new multilayered pcp and the hardness of hypergraph vertex cover. In: STOC 2003: Proceedings of the thirty-fifth annual ACM symposium on Theory of computing, pp. 595–601. ACM, New York (2003)
9. Feige, U., Jain, K., Mahdian, M., Mirrokni, V.S.: Robust combinatorial optimization with exponential scenarios. In: Fischetti, M., Williamson, D.P. (eds.) IPCO 2007. LNCS, vol. 4513, pp. 439–453. Springer, Heidelberg (2007)
10. Graham, R., Lawler, E., Lenstra, J.K., Rinnooy Kan, A.H.G.: Optimization and approximation in deterministic sequencing and scheduling: A survey. Annals of Discrete Mathematics 5, 287–326 (1979)
11. Hall, L.A., Schulz, A.S., Shmoys, D.B., Wein, J.: Scheduling to minimize average completion time: off-line and on-line algorithms. Mathematics of Operations Research 22, 513–544 (1997)
12. Kasperski, A., Zieliński, P.: On the existence of an fptas for minmax regret combinatorial optimization problems with interval data. Operations Research Letters 35(4), 525–532 (2007)
13. Khot, S.: On the power of unique 2-prover 1-round games. In: IEEE Conference on Computational Complexity, p. 25 (2002)
14. Khot, S., Regev, O.: Vertex cover might be hard to approximate to within $2-\epsilon$. In: Proc. of 18th IEEE Annual Conference on Computational Complexity (CCC), pp. 379–386 (2003)
15. Kouvelis, P., Yu, G.: Robust Discrete Optimization and Its Applications. Kluwer Academic Publishers, Dordrecht (1997)
16. Lawler, E.L.: Sequencing jobs to minimize total weighted completion time subject to precedence constraints. Annals of Discrete Mathematics 2, 75–90 (1978)
17. Margot, F., Queyranne, M., Wang, Y.: Decompositions, network flows and a precedence constrained single machine scheduling problem. Operations Research 51(6), 981–992 (2003)
18. Potts, C.N.: An algorithm for the single machine sequencing problem with precedence constraints. Mathematical Programming Study 13, 78–87 (1980)
19. Schulz, A.S.: Scheduling to minimize total weighted completion time: Performance guarantees of LP-based heuristics and lower bounds. In: Cunningham, W.H., Queyranne, M., McCormick, S.T. (eds.) IPCO 1996. LNCS, vol. 1084, pp. 301–315. Springer, Heidelberg (1996)
20. Schulz, A.S.: Scheduling to minimize total weighted completion time: performance guarantees of LP-based heuristics and lower bounds. In: Cunningham, W.H., Queyranne, M., McCormick, S.T. (eds.) IPCO 1996. LNCS, vol. 1084, pp. 301–315. Springer, Heidelberg (1996)
21. Schuurman, P., Woeginger, G.J.: Polynomial time approximation algorithms for machine scheduling: ten open problems. Journal of Scheduling 2(5), 203–213 (1999)
22. Smith, W.E.: Various optimizers for single-stage production. Naval Research Logistics Quarterly 3, 59–66 (1956)