Acta Informatica 41, 83–97 (2004)
Digital Object Identifier (DOI) 10.1007/s00236-004-0150-2

Project scheduling with irregular costs: complexity, approximability, and algorithms

Alexander Grigoriev[1], **Gerhard J. Woeginger**[2]

[1] Department of Quantitative Economies, University of Maastricht, P.O. Box 616, 6200 MD Maastricht, The Netherlands
(e-mail: a.grigoriev@ke.unimaas.nl)
[2] Department of Mathematics and Computer Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
(e-mail: gwoegi@win.tue.nl)

Received: 2 February 2004 / 13 September 2004
Published online: 29 October 2004 –

Abstract. We address a generalization of the classical discrete time-cost tradeoff problem where the costs are irregular and depend on the starting and the completion times of the activities. We present a complete picture of the computational complexity and the approximability of this problem for several natural classes of precedence constraints. We prove that the problem is NP-hard and hard to approximate, even in case the precedence constraints form an interval order. For precedence constraints with bounded height, there is a complexity jump from height one to height two: For height one, the problem is polynomially solvable, whereas for height two, it is NP-hard and APX-hard. Finally, the problem is shown to be polynomially solvable if the precedence constraints have bounded width or are series parallel.

1 Introduction

Due to its practical importance, the discrete time-cost tradeoff problem for project networks has been studied in various contexts by many researchers over the last fifty years; see Kelley & Walker (1959) for an early reference. The modern treatment of this problem started with the dynamic programming approaches of Hindelang & Muth (1979) and Robinson (1975), and with an enumeration algorithm by Harvey & Patterson (1979). An up-to-date overview on the discrete time-cost tradeoff problem is Chapter 4 of the survey by Brucker, Drexl, Möhring, Neumann & Pesch (1999) or Chapter 8 of the book by Demeulemeester & Herroelen (2002). In this paper, we look at a generalization of the classical discrete time-cost tradeoff problem where the costs depend on the exact starting *and* completion times of the activities.

Statement of the problem. Formally, we consider instances that are called *projects* and that consist of a finite set $\mathcal{A} = \{A_1, \ldots, A_n\}$ of *activities* together with a partial order $\prec$ on $\mathcal{A}$. All activities are available for processing at time zero, and they must be completed before a global project deadline T. Hence, the set of possible starting and completion times of the activities is $\{0, 1, \ldots, T\}$. The set of intervals over $\{0, 1, \ldots, T\}$ (the so-called *realizations* of the activities) is denoted by $\mathcal{R} = \{(x, y) \mid 0 \le x \le y \le T\}$. For every activity A_j, there is a corresponding cost function $c_j : \mathcal{R} \to \mathbb{R}^+ \cup \{\pm\infty\}$ that specifies for every realization $(x, y) \in \mathcal{R}$ a non-negative cost $c_j(x, y)$ that is incurred when the activity is started at time x and completed at time y. A *realization* of the project is an assignment of the activities in $\mathcal{A}$ to the intervals in $\mathcal{R}$. A realization is *feasible* if it obeys the precedence constraints: For any A_i and A_j with $A_i \prec A_j$, activity A_j is not started before activity A_i has been completed. The cost of a realization is the sum of the costs of all activities in this realization. The goal is to find a feasible realization of minimum cost. This problem is called *min-cost* project scheduling with irregular costs, or min-cost PSIC for short.

A closely related problem is *max-profit* project scheduling with irregular costs, or max-profit PSIC for short. Instead of cost functions c_j for activity A_j, here we have profit functions $p_j : \mathcal{R} \to \mathbb{R}^+ \cup \{\pm\infty\}$ that specify for every realization of A_j the resulting profit. The goal is to find a feasible realization of maximum profit. Such a profit may for instance represent the cost reduction for the project, if a deadline is stretched and an activity becomes less urgent. Clearly, the min-cost and the max-profit version are polynomial time equivalent: The transformations $c_j := \text{const}_1 - p_j$ and $p_j := \text{const}_2 - c_j$ with sufficiently large constants const_1 and const_2 translate one version into the other. However, the two versions seem to behave quite differently with respect to their approximability.

Special cases and related problems. Various special cases arise if the cost and profit functions satisfy additional properties. A cost function c is *monotone*, if $[x_1, y_1] \subseteq [x_2, y_2]$ implies $c(x_1, y_1) \ge c(x_2, y_2)$. A profit function p is *monotone*, if $[x_1, y_1] \subseteq [x_2, y_2]$ implies $p(x_1, y_1) \le p(x_2, y_2)$. The intuition behind these concepts is that short and quick executions should be more expensive than long and slow executions. It is readily seen that the general version of PSIC is equivalent to the monotone version with respect to computational complexity and approximability.

Another interesting special case arises, if $y_1 - x_1 = y_2 - x_2$ implies $c(x_1, y_1) = c(x_2, y_2)$ and $p(x_1, y_1) = p(x_2, y_2)$. In this special case, the cost and the profit of an activity only depend on the length of its realization.

This special case actually is equivalent to the DEADLINE problem for the discrete time-cost tradeoff problem: The deadline T is hard, and the goal is to assign lengths to activities such that the overall cost is minimized. Only recently, De, Dunne, Gosh & Wells (1997) proved that this problem is NP-hard in the strong sense. Skutella (1998) gives some positive approximability results, and Deineko & Woeginger (2001) give some inapproximability results for bicriteria versions. All negative results in this paper are proved for the DEADLINE problem, the weakest variant of PSIC. All positive results in this paper are proved for the most general version of PSIC.

In another special case, for every activity A_j there is a number L_j such that $c_j(x, y) < \infty$ if and only if $y - x = L_j$. In other words, activity A_j must be realized by an interval of length exactly L_j. This special case is classical project scheduling with fixed processing times. Chang & Edmonds (1985) proved that this case is polynomial time equivalent to the min-cut problem in graphs; hence, this case is polynomially solvable. Project scheduling with fixed processing times and some of its variants were also studied by Maniezzo & Mingozzi (1999) and by Möhring, Schulz, Stork & Uetz (2001).

Our results. We derive several positive and negative statements on the complexity and the approximability of min-cost and max-profit PSIC for several natural classes of precedence constraints. Our results are the following:

(1) Interval orders (Section 2). The min-cost and the max-profit version of the DEADLINE problem (and of their PSIC generalizations) are NP-hard and inapproximable even for interval orders. We establish a close (approximation preserving) connection of the min-cost DEADLINE problem to minimum vertex cover and of the max-profit DEADLINE problem to maximum independent set. All inapproximability results for these graph problems carry over to the DEADLINE problems. As an immediate consequence, unless P=NP the min-cost DEADLINE problem can not have a polynomial time approximation algorithm with worst case ratio strictly better than $7/6$. This is quite an improvement over an earlier inapproximability result of Deineko & Woeginger (2001) that only established APX-hardness for this problem.

(2) Orders of bounded height (Section 3). If the height of the precedence constraints is bounded by 2, then the DEADLINE problems and its PSIC generalizations are NP-hard and inapproximable. However, if the height of the precedence constraints is bounded by 1, then min-cost and max-profit PSIC both can be solved in polynomial time. The main idea is to translate these project scheduling problems into a maximum weight independent set problem in an underlying vertex-weighted bipartite graph.

(3) Orders of bounded width (Section 4). If the width of the precedence constraints is bounded by some fixed constant d, then min-cost and max-profit

PSIC both can be solved in polynomial time $O(n^d T^{2d+1})$. The algorithm is based on simple dynamic programming over the time axis, but the details are somewhat messy.

(4) Series parallel orders (Section 5). For series parallel precedence constraints, min-cost and max-profit PSIC can be solved in polynomial time $O(nT^3)$ by dynamic programming. This result builds on the approaches of Frank, Frisch, van Slyke & Chou (1970) and Rothfarb, Frank, Rosenbaum, Steiglitz & Kleitman (1970) for the classical discrete time-cost tradeoff problem, and extends them to the more general problems max-profit and min-cost PSIC.

(5) Finally in Section 6, we discuss how the complexity of min-cost and max-profit PSIC depends on the encoding of the input. We present an example of PSIC with two activities A and B, with the precedence constraint $A \prec B$, and with (very) specially defined cost/profit functions. For this example, even the DEADLINE problem is NP-hard.

Technical remarks. For costs and profits we allow any values from $\mathbb{R}^+ \cup \{\pm\infty\}$, that is the non-negative numbers together with plus/minus infinity. This should be seen as a useful and simple convention for specifying the input: Whenever a cost equals $+\infty$ or a profit equals $-\infty$, then the corresponding realization is forbidden. Of course this convention leads to instances that do not have any feasible realization with finite cost or profit, but these instances are easily recognized and singled out in polynomial time by the following greedy algorithm: *"In every step, select a yet unrealized activity A for which all predecessors have already been realized. Choose for A the realization (x, y) of finite cost (respectively, finite profit) with smallest value y."* This algorithm gets stuck if and only if there is no project realization of finite cost (respectively, finite profit).

Hence, throughout the paper we will restrict ourselves to instances that allow at least one realization in which all costs (respectively, all profits) are non-negative reals. A more compact representation of the input only specifies those realizations of activities that have finite costs/profits.

2 Interval orders

In this section we will derive a number of negative results for problem PSIC under interval orders. An *interval order* on a set $\mathcal{A} = \{A_1, \ldots, A_n\}$ is specified by a set of n intervals $I_1, \ldots, I_n$ along the real line. Then $A_i \to A_j$ holds if and only if the interval I_i lies completely to the left of the interval I_j, or if the right endpoint of I_i coincides with the left endpoint of I_j. See e.g. Möhring (1989).

The central proof in this section will be done by a reduction from the NP-hard INDEPENDENT SET problem in graphs; see Garey & Johnson (1979): Given a graph $G = (V, E)$ and a bound z, does G contain an independent set (a set that does not induce any edges) of cardinality z? Without loss of generality, we assume that $V = \{1, \ldots, q\}$.

We construct a project with deadline $T = 3q$ for max-profit PSIC. This project contains the activities listed below. For every activity A, we define a so-called *crucial* interval $I(A)$ that will be used to specify the interval order.

- For every vertex $i \in V$, there is a corresponding activity A_i. If A_i is realized by an interval of length zero, then its profit is $-\infty$; for an interval of length 1 or 2 the profit is 0, and for any longer realization the profit is 1. The crucial interval $I(A_i)$ for A_i is $[3i - 3, 3i]$.
- For every edge $\langle i, j \rangle \in E$ with $i < j$, there is a corresponding activity $A_{i,j}$. If $A_{i,j}$ is realized by an interval of length $3j - 3i - 2$ or more then its profit is 0, and for shorter intervals its profit is $-\infty$. The crucial interval $I(A_{i,j})$ is $[3i, 3j - 3]$.
- For $t = 0, \ldots, q$ there are so-called *blocking* activities B_t and C_t. If they are executed for at least $3t$ time units, then they bring profit 0, and for shorter intervals they bring profit $-\infty$. The crucial intervals for them are $I(B_t) = [0, 3t]$ and $I(C_t) = [3q - 3t, 3q]$.

The precedence constraints among these activities are defined as follows: For activities X and Y, $X \prec Y$ holds if and only if the crucial interval $I(X)$ lies completely to the left of the crucial interval $I(Y)$, or if the right endpoint of $I(X)$ coincides with the left endpoint of $I(Y)$. Note that this yields an interval order on the activities. Moreover, for every edge $\langle i, j \rangle \in E$ with $i < j$ this implies $A_i \prec A_{i,j} \prec A_j$.

Lemma 2.1. *If the graph G has an independent set W, then the constructed project has a feasible realization with profit $|W|$.*

Proof. Let $W \subseteq V$ denote the independent set of cardinality z. If $i \in W$, then process activity A_i with profit 1 during $[3i - 3, 3i]$. If $i \notin W$, then process it with profit 0 during $[3i-2, 3i-1]$. All other activities are processed at profit 0: Every blocking activity is processed during its crucial interval. For an edge $\langle i, j \rangle \in E$ with $i < j$ and $i \notin W$, process activity $A_{i,j}$ during $[3i - 1, 3j - 3]$; this puts $A_{i,j}$ after A_i and before A_j exactly as imposed by the precedence constraints. For an edge $\langle i, j \rangle \in E$ with $i < j$ and $i \in W$, process activity $A_{i,j}$ during $[3i, 3j - 2]$. Since $i \in W$, its neighbor j cannot be also in W; hence A_j is processed during $[3j - 2, 3j - 1]$ and after $A_{i,j}$, exactly as imposed by $A_i \prec A_{i,j} \prec A_j$.

Since in this realization activity A_i brings profit 1 if and only if $i \in W$, this realization has profit $|W|$. Moreover it can be verified that all precedence constraints indeed are satisfied. $\square$

Lemma 2.2. *If the constructed project has a feasible realization with profit $p \geq 1$, then the graph G has an independent set W with $|W| = p$.*

Proof. We first establish three simple claims on such a feasible project realization. The first claim is that (in any feasible realization with positive profit) the processing of every blocking activity must exactly occupy its crucial interval. Indeed, consider the activities B_t and C_{q-t} with their crucial intervals $I(B_t) = [0, 3t]$ and $I(C_t) = [3t, 3q]$. Since the total profit is positive, B_t is processed for at least $3t$ and C_{3q-t} is processed for at least $3q - 3t$ time units. Since B_t is a predecessor of C_{q-t}, they together cover the whole time horizon $[0, 3q]$; this fixes them in their crucial intervals.

The second claim is that every activity A_i is processed somewhere within its crucial time interval $[3i-3, 3i]$. By our first claim activity B_{i-1} completes at time $3i - 3$ and activity C_{q-i} starts at time $3i$. Since $B_{i-1} \prec A_i \prec C_{q-i}$, activity A_i cannot start before time $3i - 3$ and cannot end after time $3i$.

The third claim is that there exist exactly p activities A_i that exactly occupy their crucial intervals. By construction of the project all the profit results from the activities A_i, and A_i brings positive profit only in case it is executed for at least three time units. By our second claim, A_i cannot be executed for more than three time units. Hence, each activity A_i that brings positive profit occupies its crucial interval $[3i - 3, 3i]$.

Now we are ready to prove the statement in the lemma. Consider the set $W \subseteq V$ that contains vertex i if and only if A_i occupies its crucial interval $[3i - 3, 3i]$. We claim that W is an independent set. Suppose otherwise, and consider $i, j \in W$ with $i < j$ and $\langle i, j \rangle \in E$. Then A_i occupies $[3i - 3, 3i]$, and A_j occupies $[3j - 3, 3j]$, and $A_i \prec A_{i,j} \prec A_j$ holds. Hence, $A_{i,j}$ is processed during the $3j - 3i - 3$ time units between $3i$ and $3j - 3$. But in this case its profit is $-\infty$, and we get the desired contradiction. Hence, W is an independent set, and by our third claim $|W| = p$. □

Theorem 2.3. *Max-profit project scheduling with irregular costs is NP-hard even for interval order precedence constraints. For any $\varepsilon > 0$, the existence of a polynomial time approximation algorithm for max-profit* PSIC *for projects with n activities*

- *with worst case ratio $O(n^{1/4-\varepsilon})$ implies P=NP,*
- *with worst case ratio $O(n^{1/2-\varepsilon})$ implies ZPP=NP.*

Proof. NP-hardness follows from the Lemmas 2.1 and 2.2. The constructed reduction preserves objective values. It translates graph instances with independent sets of size z into project instances with realizations of profit z, and thus it is approximation preserving in the strongest possible sense. For a graph with q vertices, the corresponding project consists of $O(q^2)$ activities. Håstad (1999) proved that the clique problem in n-vertex graphs (and hence also the independent set problem in the complement of n-vertex graphs) cannot have a polynomial time approximation algorithm with worst case guarantee $O(n^{1/2-\varepsilon})$ unless P=NP, and it cannot have a polynomial time approximation algorithm with worst case guarantee $O(n^{1-\varepsilon})$ unless ZPP=NP. Since the blow-up in our construction is only quadratic, the theorem follows. □

In the VERTEX COVER problem, the goal is to find a minimum cardinality vertex cover (a subset of the vertices that touches every edge) for a given input graph. Note that vertex covers are the complements of independent sets. We denote by τ_{VC} the approximability threshold for the vertex cover problem, i.e., the infimum of the worst case ratios over all polynomial time approximation algorithms for this problem. Håstad (1999) proved that $\tau_{VC} \geq 7/6$ unless P=NP, and it is widely believed that $\tau_{VC} = 2$.

Theorem 2.4. *Min-cost project scheduling with irregular costs is NP-hard even for interval order precedence constraints. The existence of a polynomial time approximation algorithm for min-cost* PSIC *with worst case ratio better than τ_{VC} would imply P=NP.*

Proof. By a slight modification of the above construction. For activities $A_{i,j}$ and for blocking activities, we replace low profit $-\infty$ by high cost ∞, and the neutral profit 0 by the neutral cost 0. For activities A_i, we replace low profit $-\infty$ by high cost ∞, profit 0 by cost 1, and profit 1 by cost 0. It can be shown that there exists a realization of cost c for the constructed project, if and only if there exists an independent set of size $q - c$ for the graph, if and only if there exists a vertex cover of size c for the graph. Hence, this reduction preserves objective values. □

Corollary 2.5. *For the discrete time/cost tradeoff problem, the existence of a polynomial time approximation algorithm with worst case ratio better than τ_{VC} for the DEADLINE problem would imply P=NP.* □

3 Orders of bounded height

In this section we will derive a positive result for the project scheduling problem with irregular costs under orders of bounded height. The *height* of an ordered set is the number of elements in the longest chain minus one. Precedence constraints of height 1 are sometimes also called *bipartite* precedence constraints; see e.g. Möhring (1989).

Theorem 3.1. *Max-profit and min-cost project scheduling with irregular costs are NP-hard and APX-hard even when restricted to precedence constraints of height two.*

Proof. Deineko & Woeginger (2001) establish APX-hardness for the min-cost DEADLINE version of the discrete time/cost tradeoff problem. Their reduction produces instances of height 2 for min-cost PSIC, and it is straightforward to adapt the construction to max-profit PSIC. □

In the rest of this section we will concentrate on the max-profit PSIC for precedence constraints of height 1, and we will derive a polynomial time algorithm for it. Consider such an instance where all profits are either $-\infty$ or non-negative, and classify the activities into two types. The *A-activities* $A_1, \ldots, A_a$ do not have any predecessors, and the *B-activities* $B_1, \ldots, B_b$ do not have any successors. The only precedence constraints are of the type $A_i \to B_j$, that is from A-activities to B-activities. We start with a preprocessing phase that simplifies this instance somewhat.

- If there exists some activity that neither has a predecessor nor a successor, it is completely independent from the rest of the instance. We process this activity at the maximum possible profit, and remove it from the instance. From now on we assume that each activity has at least one predecessor or successor, and that consequently the partition into A-activities and B-activities is unique.
- We remove all realizations with profit $-\infty$ from the instance.
- Assume that there is an A-activity A_i with profit function p_i, and that there are two realizations (x, y) and (u, v) for it with $y \le v$ and $p_i(x, y) \ge p_i(u, v)$. Then the realization (x, y) imposes less restrictions on the successors of A_i and at the same time it comes at a higher profit; so we may disregard this realization (u, v) for A_i. By a symmetric argument, we may clean up the realizations of any B-activity B_j.
- Assume that $A_i \prec B_j$ and that there exists a realization (x, y) of A_i that collides with all surviving realizations of B_j (that is, the endpoint y lies strictly to the right of all possible starting points of B_j). Then we remove realization (x, y) for A_i, since it will always collide with the realization of B_j. Symmetrically, we clean up the realizations of the B-activities.

Lemma 3.2. *(i) The original instance has a realization with profit p if and only if the preprocessed instance has a realization with profit p.*

(ii) The surviving realizations for A_i can be enumerated as $(x_i^1, y_i^1), \ldots, (x_i^{a(i)}, y_i^{a(i)})$ such that they are ordered by strictly increasing right endpoint and simultaneously by strictly increasing profit for A_i. Similarly, the surviving realizations for B_j can be enumerated as $(u_j^1, v_j^1), \ldots, (u_j^{b(j)}, v_j^{b(j)})$ such that they are ordered by strictly decreasing left endpoint and simultaneously by strictly increasing profit for B_j.

(iii) If the original instance has a realization with non-negative profit, then for every activity A_i (respectively, B_j) there exists a realization in the pre-

processed instance that does not collide with any realization of a successor of A_i (respectively, of a predecessor of B_j).

Proof. Statements (i) and (ii) are clear from the preprocessing. To see (iii), consider the realization (x_i^1, y_i^1) that has the smallest right endpoint over all realizations of A_i. Suppose that it collides with some realization (u_j^ℓ, v_j^ℓ) of some successor B_j of A_i. Then this realization of B_j collides with *all* realizations of A_i and would have been removed in the last step of the preprocessing. □

From now on we assume that the conditions in (iii) in Lemma 3.2 are satisfied. We translate the preprocessed instance into a bipartite graph with weights on the vertices. The max-profit problem will boil down to finding an independent set of maximum weight in this bipartite graph.

- For every realization (x_i^k, y_i^k) of A_i with profit function p_i, there is a corresponding vertex A_i^k in the bipartite graph. If $k = 1$, then the weight of A_i^k equals $p_i(x_i^1, y_i^1)$. If $k \geq 2$, then the weight of A_i^k equals $p_i(x_i^k, y_i^k) - p_i(x_i^{k-1}, y_i^{k-1})$. Note that all weights are non-negative and that the weight of the first k realizations of A_i equals $p_i(x_i^k, y_i^k)$.
- Symmetrically, the bipartite graph contains for every realization (u_j^ℓ, v_j^ℓ) of activity B_j a corresponding vertex B_j^ℓ. The (non-negative) weights of the vertices B_j^ℓ are defined symmetrically to those of the vertices A_i^k.
- Finally, we put an edge between A_i^k and B_j^ℓ if and only if $A_i \prec B_j$ holds and if the interval $[x_i^k, y_i^k]$ does not lie completely to the left of the interval $[u_j^\ell, v_j^\ell]$.

Lemma 3.3. *The profit p of the most profitable realization of the preprocessed project equals the weight of the maximum weighted independent set in the bipartite graph.*

Proof. (Only if) Consider the most profitable realization, and consider the following set S of vertices. If activity A_i is realized as (x_i^k, y_i^k), then put the vertices $A_i^1, A_i^2, \ldots, A_i^k$ into S. The weight of these k vertices equals the profit $p_i(x_i^k, y_i^k)$ of realization (x_i^k, y_i^k). If B_j is realized as (u_j^ℓ, v_j^ℓ), then put the vertices $B_j^1, \ldots, B_j^\ell$ into S. The weight of these ℓ vertices equals the profit of the realization of B_j. By construction, the total weight of S equals the total profit p of the considered realization. Moreover, the set S is independent: If in S some A_i^s was adjacent to B_j^t, then $A_i \prec B_j$ and A_i^k and B_j^ℓ would be adjacent. But this would yield a collision in the execution of A_i and B_j, and the realization would be infeasible.

(If) Consider an independent set S of maximum weight in the bipartite graph. For an activity A_i, consider the intersection of S with $\{A_i^1, \ldots, A_i^{a(i)}\}$. By Lemma 3.2.(iii), this intersection is non-empty. Let

k denote the largest index such that A_i^k is in S. Since the neighborhood of $A_i^1, \ldots, A_i^{k-1}$ is a subset of the neighborhood of vertex A_i^k, also these $k-1$ vertices are contained in S. Then we realize activity A_i by (x_i^k, y_i^k); the resulting profit $p_i(x_i^k, y_i^k)$ equals the total weight of the vertices $A_i^1, \ldots, A_i^k$ in S. For activity B_j, we symmetrically compute a realization that is based on the maximum index ℓ for which B_j^ℓ is in S. Since A_i^k and B_j^ℓ are not incident in the bipartite graph, the chosen realizations of A_i and B_j do not collide. Hence, this realization is feasible. By construction, the total profit equals the total weight of S. □

Theorem 3.4. *Max-profit and min-cost project scheduling with irregular costs are solvable in $O(n^3T^6)$ time when restricted to precedence constraints of height one.*

Proof. By Lemma 3.3, these problems are polynomial time equivalent to finding a maximum weight independent set in a bipartite graph with non-negative vertex weights. Here, the preprocessing and instance translation require $O(n^2T^4)$ time and the resulting bipartite graph has $O(nT^2)$ vertices. Using max-flow min-cut techniques, see Ahuja, Magnanti & Orlin (1993), maximum weight independent set in bipartite graphs can be solved in $O(|V|^3)$ time, where $|V|$ is a number of vertices in bipartite graph. Thus, max-profit and min-cost project scheduling with irregular costs are solvable in $O(n^3T^6)$ time when restricted to precedence constraints of height one. □

4 Orders of bounded width

In this section, we will show that if the width of the precedence constraints is bounded by some fixed constant d, then max-profit PSIC is solvable in polynomial time. For technical reasons, we assume throughout this section that all realizations of length 0 have profit $-\infty$ and hence are forbidden; all our arguments would also go through without this assumption, but the presentation would become more complicated.

In an ordered set, two elements A_i and A_j are called *incomparable* if neither A_i is a predecessor of A_j nor A_j is a predecessor of A_i. A set of tasks is an *anti-chain*, if its elements are pairwise incomparable. The *width* of the order is the cardinality of its largest anti-chain. A well-known theorem of Dilworth (1950) states that if the width of an ordered set with n elements equals d, then this set can be partitioned into d totally ordered chains $C_1, \ldots, C_d$. Moreover, it is straightforward to compute such a chain partition in $O(n^d)$ time.

For a given instance of max-profit PSIC of width d, we first compute a chain partition $C_1, \ldots, C_d$, and we denote the number of activities in chain

C_j by n_j ($j = 1, \ldots, d$). Now let us consider some feasible realization of the project, and let us look at some fixed moment $t + \frac{1}{2}$ in time with $0 \le t \le T$. As the chain C_j is totally ordered, at time $t + \frac{1}{2}$, at most one of its activities is under execution. Chain C_j is called *inactive* at time $t + \frac{1}{2}$ if none of its activities is under execution, and otherwise it is *active* at time $t + \frac{1}{2}$.

Definition 4.1. *For a feasible realization, the* snapshot S *taken at time* $t + \frac{1}{2}$ *with* $0 \le t \le T$ *contains the following information:*

(S1) For every chain C_j, one bit of information that specifies whether C_j is active or inactive.
(S2) For every inactive chain C_j, a number IN_j *with* $0 \le \text{IN}_j \le n_j$ *that specifies the last activity in C_j that was executed before time $t + \frac{1}{2}$. If no activity has been executed so far, then* $\text{IN}_j = 0$.
(S3) For every active chain C_j, a number ACT_j *with* $1 \le \text{ACT}_j \le n_j$ *that specifies the current activity of C_j. Moreover, the starting time x_j of the current activity with $0 \le x_j \le T - 1$.*

For the data in (S1) there are at most 2^d possibilities, for all the numbers IN_j and ACT_j in (S2) and (S3) there are at most $O(n^d)$ possibilities, and for all the starting times in (S3) there are at most $O(T^d)$ possibilities. Since d is a fixed constant, this yields that there are at most $O(n^d T^d)$ snapshots at time $t + \frac{1}{2}$.

Definition 4.2. *For any t with $0 \le t \le T$ and for any possible snapshot S, we denote by $F[t; S]$ the maximum possible profit that can be earned on activities completing before time $t + \frac{1}{2}$ in a feasible project realization whose snapshot at time $t + \frac{1}{2}$ equals S.*

If no such feasible realization exists, then $F[t; S] = -\infty$.

We compute all these values $F[t; S]$ by a dynamic programming approach that works through them by increasing t. The initial cases with $t = 0$ are trivial, since $F[0; S]$ can only take the values 0 (if there exists a feasible realization with snapshot S at time $\frac{1}{2}$) or $-\infty$ (otherwise). To compute $F[t; S]$ for $t \ge 1$, we check all possibilities for a compatible predecessor snapshot S' at time $t - \frac{1}{2}$ in the following way by considering all the chains separately (the data from snapshots S and S' is represented by un-primed and by primed variables, respectively):

– Chain C_j might be active in S' and inactive in S. Then $\text{IN}_j = \text{ACT}'_j$. The additional profit comes from realizing the ACT'_j-th activity in chain C_j from time x'_j to time t.
– Chain C_j might be inactive in S' and active in S. Then $\text{ACT}_j = \text{IN}'_j + 1$ and $x_j = t$. No additional profit is generated.

- Chain C_j might be inactive in S' and S. Then $\text{IN}_j = \text{IN}'_j$. Since no activity can simultaneously be started and completed at time t, no additional profit is generated.
- Chain C_j might be active in S' and S. There are two cases: If the same activity is executed at time $t - \frac{1}{2}$ and at time $t + \frac{1}{2}$, then $\text{ACT}_j = \text{ACT}'_j$ and $x_j = x'_j$, and no additional profit is generated. And if the executed activities at times $t - \frac{1}{2}$ and $t + \frac{1}{2}$ are distinct, then $\text{ACT}_j = \text{ACT}'_j + 1$ and $x_j = t$ must hold. The additional profit comes from realizing the ACT'_j-th activity in C_j from time x'_j to time t.

If snapshots S and S' are of this form for all d chains, then we say that S' is a *predecessor* of S. Moreover, we denote the total additionally generated profit over all the chains by profit(S', S). It can be verified that any snapshot S at time $t+\frac{1}{2}$ has at most $O(T^d)$ predecessors at time $t-\frac{1}{2}$. Then the value $F[t; S]$ can be computed as

$$F[t;S] := \max\{F[t-1; S'] + \text{profit}(S', S) \mid S' \text{ is a predecessor of } S\}. \quad (1)$$

In the end, the solution to the instance of max-profit PSIC can be found in $F[T; S^*]$ where S^* is the snapshot at time $T+\frac{1}{2}$ where all chains are inactive and where $\text{IN}_j = n_j$ holds for $j = 1, \dots, d$. The time complexity of this dynamic programming algorithm is $O(n^d T^{2d+1})$: Since there are $O(n^d T^d)$ snapshots at time $t + \frac{1}{2}$, we altogether compute $O(n^d T^{d+1})$ values $F[t; S]$. Each value can be computed in $O(T^d)$ time by checking all predecessors in (1). By storing appropriate auxiliary information and by performing some backtracking, one can also explicitly compute the optimal feasible realization while increasing the running time only by a constant factor. Since these are standard techniques, we do not elaborate on them.

Theorem 4.3. *Max-profit and min-cost project scheduling with irregular costs are polynomially solvable in $O(n^d T^{2d+1})$ time when restricted to precedence constraints of width bounded by the fixed constant d.* □

5 Series parallel orders

Precedence constraints are called *series parallel* if (i) they contain a single vertex, or (ii) they form the series composition of two series parallel order, or (iii) they form the parallel composition of two series parallel orders. Only orders that can be constructed via rules (i)–(iii) are series parallel. Here the *series composition* of two orders $(V_1, \prec_1)$ and $(V_2, \prec_2)$ with $V_1 \cap V_2 = \emptyset$ is the order that results from taking their union and making all elements in V_1 predecessors of all elements in V_2, whereas the *parallel composition* of $(V_1, \prec_1)$ and $(V_2, \prec_2)$ simply is their disjoint union. Series parallel precedence constraints are a proper generalization of tree precedence constraints; see e.g. Möhring (1989).

It is well known that a series parallel order can be decomposed in polynomial time into its atomic parts according to the series and parallel compositions; see e.g. Valdes, Tarjan & Lawler (1982). Essentially, such a decomposition corresponds to a rooted, ordered, binary tree where all interior vertices are labeled by s or p (series or parallel composition) and where all leaves correspond to single elements of the order. We associate with every interior vertex v of the decomposition tree the series parallel order SP(v) that is induced by the leaves of the subtree below v. Note that for the root vertex *root* of the decomposition tree, the corresponding order SP(*root*) is the whole ordered set.

Our goal is to design a polynomial time algorithm for max-profit PSIC with series parallel precedence constraints. The usual tool for dealing with series parallel structures is dynamic programming.

Definition 5.1. *For a vertex v in the decomposition tree, and for integers x and y with $0 \leq x \leq y \leq T$, we denote by $F[v; x, y]$ the maximum possible profit that can be earned on the activities in* SP(v)*, subject to the condition that all these activities are executed somewhere during the time interval $[x, y]$ such that they obey the precedence constraints.*

If no such feasible realization exists, then $F[v; x, y] = -\infty$.

We compute all these values $F[v; x, y]$ by a dynamic programming approach that starts in the leaves of the decomposition tree, and then moves upwards towards the root.

- If v is a leaf, the order SP(v) consists of a single activity A, and $F[v; x, y]$ is easily computed.
- If v is a p vertex with left child v_1 and right child v_2, then $F[v; x, y] := \; F[v_1; x, y] + F[v_2; x, y]$
- If v is an s vertex with left child v_1 and right child v_2, then $F[v; x, y] := \; \max\{F[v_1; x, z] + F[v_2; z, y] : \; x \leq z \leq y\}$

In the end, the solution to the instance of max-profit PSIC can be found in $F[root; 0, T]$. The time complexity of this dynamic programming algorithm is $O(nT^3)$: To compute the values $F[v; x, y]$ for the $O(nT^2)$ leaves, it is sufficient to look once at every possible realization of every activity; this altogether costs $O(nT^2)$ time. And for the inner vertices v, the corresponding $O(nT^2)$ values can be computed in $O(T)$ time per value. By standard techniques, one can also explicitly compute the optimal feasible realization while increasing the running time only by a constant factor.

Theorem 5.2. *Max-profit and min-cost project scheduling with irregular costs are polynomially solvable in $O(nT^3)$ time when restricted to series parallel precedence constraints.* □

6 PSIC with compactly encoded inputs

In all the sections above, we assumed that the cost and profit functions are specified *pointwise*, that is, that the input lists for every possible realization $(x, y) \in \mathcal{R}$ of every project the corresponding non-negative cost, respectively the corresponding non-negative profit. In this section, we briefly discuss the variant where the cost and profit functions can be encoded *compactly* via a fast oracle algorithm.

We present a pathological example for the min-cost version of this variant; a pathological example for the max-profit version can be derived in a similar fashion.

Theorem 6.1. *The special case of the DEADLINE problem with only two activities $A \prec B$ and with compactly encoded cost functions is NP-hard in the ordinary sense.*

Proof. The proof is done by a reduction from the NP-hard THREE-SATISFIABILITY problem; see Garey & Johnson (1979): Given a collection $C = \{c_1, c_2, \ldots, c_m\}$ of clauses over a finite set $U = \{x_1, x_2, \ldots, x_n\}$ of logical variables such that every clause contains exactly three literals, does there exist a truth assignment for U that satisfies all the clauses in C?

With every n-bit integer F with bits $f_1, f_2, \ldots, f_n$, we associate a corresponding truth assignment for the variables $x_1, x_2, \ldots, x_n$ that sets x_k =TRUE if $f_k = 1$, and x_k =FALSE if $f_k = 0$. Consider the following instance of the DEADLINE problem with time horizon $T = 2^n$, and with two activities A and B where $A \prec B$:

- If activity A is realized at a length of ℓ with $0 \le \ell \le T$, then the resulting cost $c_A(\ell)$ equals $2T - 2\ell$ if the true assignment corresponding to ℓ satisfies the given THREE-SATISFIABILITY instance, and otherwise the cost equals $2T - 2\ell + 1$.
- For any ℓ with $0 \le \ell \le T$, the cost $c_B(\ell)$ of realizing activity B at a length of ℓ equals $2T - 2\ell$.

Note that the defined cost functions are strictly decreasing in ℓ. The cost function c_A is compactly encoded via the clause set C, and for any given value ℓ it can be evaluated in polynomial time. If there is a satisfying truth assignment, then the optimal cost is $2T$. If there is no satisfying truth assignment, then the optimal cost is $2T + 1$. $\square$

Acknowledgements. We thank Yuri Kochetov, Alexander Kononov, and Maxim Sviridenko for several helpful discussions and comments on the topic of this paper.

References

1. Ahuja, R.K., Magnanti, T.L., Orlin, J.B. (1993) Network flows: Theory, algorithms, and applications. Prentice Hall, Upper Saddle River, NJ
2. Brucker, P., Drexl, A., Möhring, R., Neumann, K., Pesch, E. (1999) Resource-constrained project scheduling: Notation, classification, models, and methods. European Journal of Operational Research 112: 3–41
3. Chang, G.J., Edmonds, J. (1985) The poset scheduling problem. Order 2: 113–118
4. De, P., Dunne, E.J., Gosh, J.B., Wells, C.E. (1995) The discrete time-cost tradeoff problem revisited. European Journal of Operational Research 81: 225–238
5. De, P., Dunne, E.J., Gosh, J.B., Wells, C.E. (1997) Complexity of the discrete time-cost tradeoff problem for project networks. Operations Research 45: 302–306
6. Deineko, V.G., Woeginger, G.J. (2001) Hardness of approximation of the discrete time-cost tradeoff problem. Operations Research Letters 29: 207–210
7. Demeulemeester, E.L., Herroelen, W.S. (2002) Project scheduling: A research handbook. Kluwer Academic Publishers
8. Dilworth, R.P. (1950) A decomposition theorem for partially ordered sets. Annals of Mathematics 51: 161–166
9. Frank, H., Frisch, I.T., van Slyke, R., Chou, W.S. (1970) Optimal design of centralized computer networks. Networks 1: 43–58
10. Garey, M.R., Johnson, D.S. (1979) Computers and intractability: A guide to the theory of NP-completeness. Freeman, San Francisco
11. Harvey, R.T., Patterson, J.H. (1979) An implicit enumeration algorithm for the time/cost tradeoff problem in project network analysis. Foundations of Control Engineering 4: 107–117
12. Håstad, J. (1997) Some optimal inapproximability results. Proceedings of the 29th ACM Symposium on the Theory of Computing (STOC'1997), pp. 1–10
13. Håstad, J. (1999) Clique is hard to approximate within $n^{1-\epsilon}$. Acta Mathematica 182: 105–142
14. Hindelang, T.J., Muth, J.F. (1979) A dynamic programming algorithm for decision CPM networks. Operations Research 27: 225–241
15. Kelley, J.E., Walker, M.R. (1959) Critical path planning and scheduling: An introduction. Mauchly Associates Inc., Ambler, PA
16. Maniezzo, V., Mingozzi, A. (1999) The project scheduling with irregular cost distribution. Operations Research Letters 25: 175–182
17. Möhring, R.H. (1989) Computationally tractable classes of ordered sets. In: Rival, I. (ed.) Algorithms and order, pp. 105–193. Kluwer Academic Publishers
18. Möhring, R.H., Schulz, A.S., Stork, F., Uetz, M. (2001) On project scheduling with with irregular starting time costs. Operations Research Letters 28: 149–154
19. Robinson, D.R. (1975) A dynamic programming solution to cost-time tradeoff for CPM. Management Science 22: 158–166
20. Rothfarb, B., Frank, H., Rosenbaum, D.M., Steiglitz, K., Kleitman, D.J. (1970) Optimal design of offshore natural-gas pipeline systems. Operations Research 18: 992–1020
21. Skutella, M. (1998) Approximation algorithms for the discrete time-cost tradeoff problem. Mathematics of Operations Research 23: 909–929
22. Valdes, J., Tarjan, R.E., Lawler, E.L. (1982) The recognition of series-parallel digraphs. SIAM Journal on Computing 11: 298–313