Complexity analysis of the discrete sequential search problem with group activities

Coolen K, Talla Nobibon F, Leus R.

KBI_1313

Complexity analysis of the discrete sequential search problem with group activities

Kris Coolen
Department of Decision Sciences and Information Management, KU Leuven, Belgium

Fabrice Talla Nobibon
Department of Decision Sciences and Information Management, KU Leuven, Belgium

Roel Leus[1]
Department of Decision Sciences and Information Management, KU Leuven, Belgium
E-mail: Roel.Leus@kuleuven.be
Postal address: ORSTAT, Faculty of Business and Economics, KU Leuven, Naamsestraat 69, B-3000 Leuven, Belgium
Tel. +32 16 32 69 67. Fax +32 16 32 66 24

[1]Corresponding author

Abstract. We study the computational complexity of the discrete sequential search problem with group activities, in which a set of boxes and group activities is given, and a single object is hidden in one of these boxes. The goal is to find a sequence in which the boxes are to be searched and the group activities will be executed to minimize the expected total cost of finding the object while satisfying the precedence constraints imposed by the group activities. We prove that this problem is strongly NP-hard both for conjunctive group activities and for disjunctive group activities, and we discuss some special cases that can be solved in polynomial time.

Keywords: discrete sequential search; group activities; NP-hard; single-machine scheduling; bipartite precedence constraints.

1 Introduction

The purpose of this paper is to study the computational complexity of the *discrete sequential search problem with group activities* as defined by Wagner and Davis [16]. A single object is hidden in one of n boxes and the probability that box k contains that object is π_k ($\sum_{k=1}^{n} \pi_k = 1$). The boxes are searched one at a time and the cost of searching box k is t_k. If the box containing the object is searched then the object is detected with certainty. When the first $n-1$ searches are negative, it is certain that the item is hidden in the last unsearched box. It is assumed that this final box must still be opened (and therefore its cost is incurred). There are also m 'group activities.' Each group activity ℓ has a cost R_ℓ and is associated with a subset $S_\ell \subseteq \{1,\ldots,n\}$ of boxes. Note that some boxes may appear in more than one subset S_ℓ. The group activities are said to be *conjunctive* if any box can be searched only when all the group activities in which it appears have been performed, whereas for *disjunctive* group activities a box can be searched as soon as at least one of the group activities in which it appears has been executed. The goal is to find a sequence (defining a search strategy) in which the boxes are to be searched and the group activities are to be performed so as to minimize the expected cost while satisfying the precedence constraints imposed by the group activities. We refer to the discrete sequential search problem with exclusively conjunctive, respectively disjunctive, group activities as Problem 1, respectively Problem 2.

Discrete sequential search problems have applications in various areas such as quality control [11], research and development [9], and diagnostic tests on components of complex radar, missile, and communications systems [6]. In the diagnostic sequencing problem, illustrations of a group activity include removing an access cover, draining fluids, disconnecting a power supply, etc., which must occur before a set of components can be tested. In the problem of sequencing tasks in a research-and-development project, a group activity may represent a facility that must be constructed before a set of tasks can be completed [16].

In 2001, Wagner and Davis [16] presented an integer-programming model for the discrete sequential search problem including both conjunctive and disjunctive group activities. Based on their experiments, they conjectured that the conjunctive case (Problem 1) could be solved as a linear programming problem. Recently in [15], we have described a counterexample for which the optimal value of the linear program proposed by Wagner and Davis is different from the optimal value of the integer-programming model, hence contradicting their conjecture.

In this paper, we first observe in Section 2 that Problem 1, respectively Problem 2, is equivalent to scheduling a set of jobs on a single machine to minimize the total weighted completion time with *and*, respectively *or*, precedence constraints, represented by a special bipartite graph. We exploit this equivalence to establish NP-hardness results for Problem 1 in Section 3, and for Problem 2 in Section 4. Subsequently, we study some polynomially solvable cases of Problem 1 and Problem 2 in Section 5. Finally, a conclusion and an outlook on further work is given in Section 6.

2 Link with the scheduling literature

Consider the problem where each of n jobs is to be processed without interruption on a single machine that can handle only one job at a time. Job i ($i = 1, \ldots, n$) becomes available at time zero (no release dates), requires a processing time p_i and has a non-negative weight w_i. The objective is to find a processing order of the jobs that minimizes the sum of weighted completion times $\sum_{i=1}^{n} w_i C_i$, where C_i is the time at which job i completes in the given schedule. In standard notation [7] the problem is referred to as $1||\sum w_i C_i$. This generic problem has an $O(n \log n)$ algorithm based on Smith's rule [14], which schedules jobs in such a way that for all pairs of jobs i and j, job i is executed before job j if $p_i w_j < p_j w_i$.

The discrete sequential search problem without group activities is equivalent to $1||\sum w_i C_i$ in the sense that any algorithm for the first problem can be used to solve the second problem, and vice versa. On the one hand, when we are given an instance of the search problem, we can construct an instance of $1||\sum w_i C_i$ if we associate with each box i a job i with weight $w_i := \pi_i$ and processing time $p_i := t_i$. In this way the total expected cost of finding the hidden object for any search order of the boxes equals the total weighted completion time of the processing order of the corresponding jobs. On the other hand, for any instance of $1||\sum w_i C_i$, we create for each job i a box i with $\pi_i := w_i / W$ and $t_i := p_i$, where $W = \sum_{i=1}^{n} w_i$. If z is the total expected search cost of some order of the boxes, then the total weighted completion time of the corresponding order of jobs is Wz. By this equivalence, it follows that the discrete sequential search problem without group activities can be solved in time $O(n \log n)$ by ordering the boxes k in non-decreasing order of t_k / π_k.

The problem $1||\sum w_i C_i$ has been extended following several directions, including the addition of precedence constraints among jobs. When precedence constraints are included, the problem is written as $1|prec|\sum w_i C_i$. In the literature, precedence constraints are specified by a directed acyclic graph $G = (J, A)$, where $J = \{1, \ldots, n\}$ and an arc $(i, j) \in A$ indicates that job i must be executed before job j. Lenstra and Rinnooy Kan [10] show that $1|prec|\sum w_i C_i$ is strongly NP-hard when G is an arbitrary directed acyclic graph, even if each job has a unit processing time. Ambühl et al. [2] prove that $1|prec|\sum w_i C_i$ remains strongly NP-hard if the precedence constraints form an interval order. Some polynomially solvable cases have been studied by Horn [8] and Sidney [13], who present an $O(n \log n)$ algorithm when G is a rooted tree, and by Adolphson [1], who describes an $O(n \log n)$ algorithm when G is a series-parallel graph. Ambühl et al. [2] exploit the relationship between the dimension theory of partial orders and $1|prec|\sum w_i C_i$ to obtain a polynomial-time 4/3-approximation algorithm when G is a convex bipartite graph or a unit interval graph, and to obtain a 3/2-approximation for an arbitrary interval graph. These approximation results improve previous results by Woeginger [17] and are the currently best-known approximation ratios.

For ease of exposition, when the precedence constraints are represented by a bipartite graph $G = (V_1 \cup V_2, A)$ (thus for each $(i, j) \in A$, $i \in V_1$ and $j \in V_2$), we write $1|V_1 \cup V_2|\sum w_i C_i$. Now consider the special case where the weights of the jobs in V_1 are zero, which is denoted by $1|V_1 \cup V_2, w(V_1) = 0|\sum w_i C_i$ (for any index set $A \subseteq \{1, \ldots, n\}$ we define $w(A) = \sum_{i \in A} w_i$). The variant of $1|V_1 \cup V_2, w(V_1) = 0|\sum w_i C_i$ in which any job in V_2 can be executed as soon as at least one of its predecessors in V_1 has been processed (*or*-type precedence constraints [5, 12]), is denoted by $1|V_1 \cup V_2, w(V_1) = 0, or\,|\sum w_iC_i$.

Lemma 1. *Problem 1 is equivalent to* $1|V_1 \cup V_2, w(V_1) = 0|\sum w_iC_i$*, and Problem 2 is equivalent to* $1|V_1 \cup V_2, w(V_1) = 0, or\,|\sum w_iC_i$*.*

Proof. Consider an instance of Problem 1 with n boxes and m group activities. We construct an equivalent instance of $1|V_1 \cup V_2, w(V_1) = 0|\sum w_iC_i$ with $m+n$ jobs, where the first m jobs, called *group-activity jobs*, correspond with the m group activities and belong to V_1 whereas the last n jobs, called *box jobs*, correspond with the n boxes and all belong to V_2. The weight w_i of group-activity job i ($i = 1, \dots, m$) is 0 whereas the weight w_i of box job i ($i = m+1, \dots, m+n$) is the probability π_{i-m} that the object is in the box $i-m$. Next, the processing time p_i of a group-activity job i ($i = 1, \dots, m$) is exactly the cost R_i of group activity i whereas the processing time p_i of box job i ($i = m+1, \dots, m+n$) is equal to the cost t_{i-m} of searching box $i-m$. Finally, the bipartite precedence graph $G = (V_1 \cup V_2, A)$ is such that there is an arc from group-activity job i ($i = 1, \dots, m$) to box job j ($j = m+1, \dots, m+n$) if and only if box $j-m$ belongs to S_i (the subset of boxes associated with group activity i). This construction is done in polynomial time with respect to the size of the instance (in this case $n+m$). We can revert this construction to build an instance of Problem 1 from a given instance of $1|V_1 \cup V_2, w(V_1) = 0|\sum w_iC_i$, but then the job weights must be scaled to guarantee that the box probabilities π_i sum to one (as explained in the second paragraph of this section). For disjunctive group activities (Problem 2), the same construction is valid, but now a job in V_2 can be started as soon as at least one of its predecessors in V_1 has been executed (*or*-type precedence constraints). $\square$

3 Complexity of Problem 1

We first observe that the algorithms developed by Adolphson [1] and Horn [8] cannot solve Problem 1 because the corresponding bipartite graph is neither always a rooted tree nor always a series-parallel graph. In this section, we prove that $1|V_1 \cup V_2, w(V_1) = 0|\sum w_iC_i$ is strongly NP-hard, even if each job in V_1 has a unit processing time and each job in V_2 has a zero processing time and a unit weight. We use a reduction from $1|prec, p_i = 1|\sum w_iC_i$, which is known to be NP-hard in the strong sense [10]. From Lemma 1, Problem 1 is then also strongly NP-hard (see Corollary 1 below).

Theorem 1. *The problem* $1|V_1 \cup V_2, w(V_1) = 0|\sum w_iC_i$ *is strongly NP-hard, even if each job in* V_1 *has a processing time of one and each job in* V_2 *has a processing time of zero and a weight of one.*

Proof. Clearly, the decision variant of $1|V_1 \cup V_2, w(V_1) = 0|\sum w_iC_i$ belongs to the class NP. Consider an arbitrary instance $\mathcal{I}$ of $1|prec, p_i = 1|\sum w_iC_i$ with job set $J = \{1, \dots, n\}$, where each $i \in J$ has a processing time $p_i = 1$ and a non-negative integer weight w_i. The precedence constraints are described by a (directed acyclic) graph $G(J, A)$. We now construct an instance of $1|V_1 \cup V_2, w(V_1) = 0|\sum w_iC_i$, which we denote by $f(\mathcal{I})$, as follows. With each job $i \in J$ we associate two subsets J_i^1 and J_i^2, consisting of $p_i = 1$ and w_i copies of job i, respectively. We define $V_1 = \cup_{i \in J} J_i^1$ and $V_2 = \cup_{i \in J} J_i^2$. For each $i \in J$, the single job in J_i^1 has a weight of 0 and a processing time of 1, whereas for each job in J_i^2 we have a weight of 1 and a processing time of 0. Furthermore, there is an arc from the only job in J_i^1 to each job in J_j^2 when either $i = j$ or $(i, j) \in T(A)$, where $T(A)$ is the transitive closure of A. Figure 1 shows the constructed instance $f(\mathcal{I})$ for an example

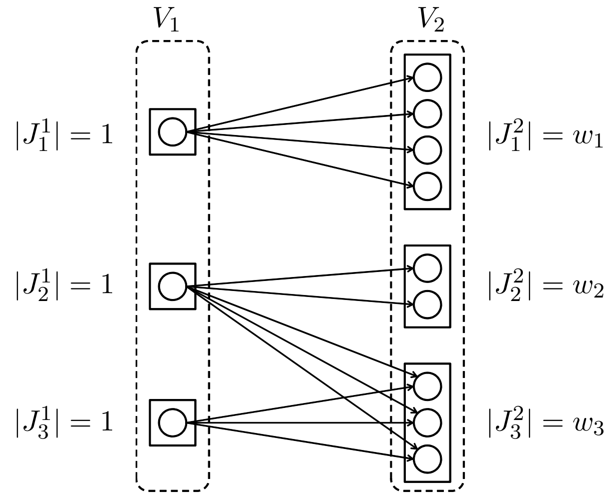

Figure 1: Constructed instance $f(\mathcal{I})$ of $1|V_1 \cup V_2, w(V_1) = 0|\sum w_i C_i$ in the proof of Theorem 1, where $\mathcal{I}$ is an instance of $1|prec, p_i = 1|\sum w_i C_i$ with $n = 3$, $w_1 = 4$, $w_2 = 2$, $w_3 = 3$, and $A = \{(2,3)\}$

instance $\mathcal{I}$ of $1|prec, p_i = 1|\sum w_i C_i$ with $n = 3$, $w_1 = 4$, $w_2 = 2$, $w_3 = 3$, and $A = \{(2,3)\}$. This completes the construction of the instance $f(\mathcal{I})$ of $1|V_1 \cup V_2, w(V_1) = 0|\sum w_i C_i$. Note that it is a pseudo-polynomial construction because it is polynomial in n and $\sum_{i \in J} w_i$. However, as $1|prec, p_i = 1|\sum w_i C_i$ is strongly NP-hard, it is sufficient to show that f is a pseudo-polynomial transformation (see [4, p. 101]).

Note that any permutation of elements in $V_1 \cup V_2$ that satisfies the precedence constraints is a feasible solution to $f(\mathcal{I})$. We now describe a subclass of feasible solutions to $f(\mathcal{I})$ that will be used to show the equivalence between $\mathcal{I}$ and $f(\mathcal{I})$: we only consider the solutions to $f(\mathcal{I})$ in which for each $i \in J$, the jobs in J_i^2 are scheduled consecutively and the single job in J_i^1 immediately precedes the block of jobs J_i^2. We call such a solution a *consecutive-index* solution. For the moment, let us assume that the following claim holds; its proof is established below.

Claim 1. *Any optimal solution to $f(\mathcal{I})$ is a consecutive-index solution.*

We now argue that any solution $(i_1, \ldots, i_n)$ to $\mathcal{I}$ can be transformed into a consecutive-index solution $(J_{i_1}^1, J_{i_1}^2, \ldots, J_{i_n}^1, J_{i_n}^2)$ to $f(\mathcal{I})$ with the same objective value, and vice versa. It can be verified that both schedules have a total weighted completion time equal to $\sum_{k=1}^n k w_{i_k}$ (recall that we have unit processing times in $\mathcal{I}$). Claim 1 together with this result will imply that $1|V_1 \cup V_2, w(V_1) = 0|\sum w_i C_i$ is at least as hard as $1|prec, p_i = 1|\sum w_i C_i$. Because the latter is strongly NP-hard, we conclude that Theorem 1 holds. □

Proof of Claim 1. We prove Claim 1 by induction on the the number of jobs $|J| = n$. When $n = 1$, all solutions to $f(\mathcal{I})$ are consecutive-index solutions, and take the form (J_1^1, J_1^2). Now assume that the claim holds whenever $n = r - 1$ with $r > 1$. In the remainder of this paragraph we will show that for any optimal solution α to $f(\mathcal{I})$ there is a job $i_1 \in J$ without predecessors in G such that the jobs in $J_{i_1}^1$ and $J_{i_1}^2$ are the first jobs in α, in that order. First note that it is a dominant decision to schedule a job of V_2 as soon as possible. Indeed, if we move a job of V_2 earlier in the schedule, we can only decrease its completion time, and since it has zero processing time, this will not increase the completion time of any other job in the schedule. As a result, jobs in any subset J_i^2 are scheduled consecutively in α. Let $J_{i_1}^2$ be the first block of jobs of V_2 in α. By construction, the job in $J_{i_1}^1$ and the jobs in any J_j^1 for which $(j, i_1) \in T(A)$, are scheduled before $J_{i_1}^2$. It can be seen that α cannot be optimal if i_1 has predecessors in G. Indeed, if i_1 has predecessors in G then there is a predecessor job j without predecessors (G is acyclic). The schedule α can then be improved by first scheduling J_j^1 and J_j^2.

We conclude that $\alpha = (J_{i_1}^1, J_{i_1}^2, \alpha')$ where α' is an optimal solution to $f(\mathcal{I}')$ with $\mathcal{I}'$ the instance of $1|prec, p_i = 1|\sum w_i C_i$ obtained from $\mathcal{I}$ by removing i_1 from J together with all outgoing arcs. The instance $\mathcal{I}'$ has only $r-1$ jobs, and the associated precedence graph is again acyclic. By induction, α' is a consecutive index solution to $f(\mathcal{I}')$, thus we can write $\alpha' = (J_{i_2}^1, J_{i_2}^2, \ldots, J_{i_r}^1, J_{i_r}^2)$ with $\{i_2, \ldots, i_r\} = J \setminus \{i_1\}$. We conclude that $\alpha = (J_{i_1}^1, J_{i_1}^2, J_{i_2}^1, J_{i_2}^2, \ldots, J_{i_r}^1, J_{i_r}^2)$ is a consecutive-index solution to $f(\mathcal{I})$. $\square$

Woeginger [17] shows that the special case of $1|V_1 \cup V_2, w(V_1) = 0|\sum w_i C_i$ described in Theorem 1 is as hard to approximate as the general case $1|prec|\sum_i w_i C_i$. It should be noted that from this result, Theorem 1 can also be inferred. From Lemma 1 the following result ensues for Problem 1, and is valid even if all group activities have a unit cost, and all boxes are identical with a zero inspection cost.

Corollary 1. *Problem 1 is strongly NP-hard, even if $R_\ell = 1$ for all group activities ℓ, and for all the n boxes i we have $\pi_i = 1/n$ and $t_i = 0$.*

Corollary 1 implies that, unless P = NP, there is no (concise) LP-model for solving Problem 1. In particular, the integrality constraints in the integer-programming model proposed in [16] cannot be relaxed, which reinforces the counterexample presented in [15].

An interesting special case of Problem 1 is the setting where a linear ordering of the boxes exists such that for each group activity ℓ, the associated boxes in S_ℓ are consecutive in this ordering. This geometrical property may be valid in several practical applications such as the presence of access covers over a set of consecutive machine components. The corresponding special case of $1|V_1 \cup V_2, w(V_1) = 0|\sum w_i C_i$ that is equivalent to this particular setting of Problem 1 is such that the bipartite precedence graph is *convex*. That is, the jobs in V_2 can be ordered such that for every job in V_1, the successor jobs in V_2 are consecutive in that ordering. The polynomial-time 4/3-approximation algorithm that was presented in [2] can be applied to approximate this special case of Problem 1. The complexity status of $1|prec|\sum_i w_i C_i$ with precedence constraints that form a convex bipartite order is still an open problem [2]. Remark that in this open problem, there is no restriction on the weights, whereas we demand zero weights for the jobs in V_1. Therefore, an NP-hardness result for the special case of $1|V_1 \cup V_2, w(V_1) = 0|\sum w_i C_i$ with convex bipartite precedence constraints would settle the open problem in [2], whereas a positive result for the former problem would not necessarily apply for the latter.

4 Complexity of Problem 2

We now consider the disjunctive case (Problem 2), which is equivalent to $1|V_1 \cup V_2, w(V_1) = 0, or\,|\sum w_i C_i$, in which any job in V_2 can be executed as soon as at least one of its predecessors in V_1 has been processed (Lemma 1). The result of Corollary 1 does not apply to Problem 2 because it assumes that a box can be executed only when all its associated group activities have been

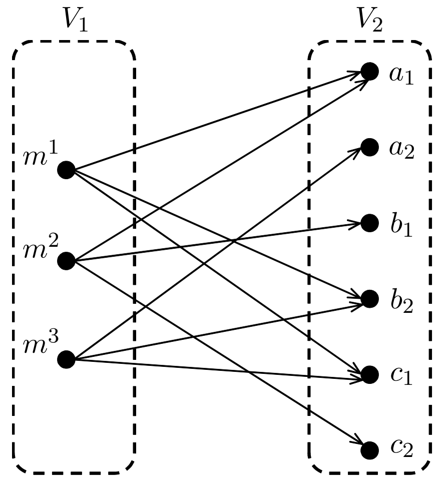

Figure 2: Constructed instance of $1|V_1 \cup V_2, w(V_1) = 0|\sum w_iC_i$ in the proof of Theorem 2 for $q = 2$ and $M = \{m^1, m^2, m^3\}$, with $m^1 = (a_1, b_2, c_1)$, $m^2 = (a_1, b_1, c_2)$ and $m^3 = (a_2, b_2, c_1)$

processed. We show that $1|V_1 \cup V_2, w(V_1) = 0, or\,|\sum w_iC_i$ is strongly NP-hard using a reduction from the variant of 3-Dimensional Matching (3DM) defined as follows [4]:

3-Dimensional Matching (3DM):
Instance: A set $M \subseteq A \times B \times C$, where A, B, and C are disjoint sets having the same number q of elements and such that each element of A, B and C is the coordinate of at least one triple in M.
Question: Does M contain a matching, that is, a subset $M' \subseteq M$ such that $|M'| = q$ and no two elements of M' contain an identical coordinate?

Theorem 2. *$1|V_1 \cup V_2, w(V_1) = 0, or\,|\sum w_iC_i$ is strongly NP-hard, even when all jobs in V_1 have unit processing times and exactly three successor jobs in V_2, and all jobs in V_2 have zero processing times and unit weights.*

Proof. Clearly, the decision variant of $1|V_1 \cup V_2, w(V_1) = 0, or\,|\sum w_iC_i$ belongs to the class NP. Consider an arbitrary instance of 3DM described by the three distinct sets $A = \{a_1, \ldots, a_q\}$, $B = \{b_1, \ldots, b_q\}$, $C = \{c_1, \ldots, c_q\}$, and $M = \{m^1, \ldots, m^{|M|}\}$, where $m^i = (a^i, b^i, c^i) \in A \times B \times C$. We build an instance of $1|V_1 \cup V_2, w(V_1) = 0, or\,|\sum w_iC_i$ with $V_1 = \{m^1, \ldots, m^{|M|}\}$ containing $|M|$ elements, each corresponding with one $m^i \in M$. The set $V_2 = \{a_1, \ldots, a_q, b_1, \ldots, b_q, c_1, \ldots, c_q\}$ contains $3q$ jobs. Each job $m^i \in V_1$ is the predecessor of jobs $a^i, b^i, c^i \in V_2$. Furthermore, each job $m^i \in V_1$ has a weight of $w_{m^i} = 0$ and a processing time of $p_{m^i} = 1$, whereas each job $e \in V_2$ has a weight of $w_e = 1$ and a processing time of $p_e = 0$. This completes the description of our instance of $1|V_1 \cup V_2, w(V_1) = 0, or\,|\sum w_iC_i$. This construction can be set up in polynomial time. Figure 2 illustrates this construction for $q = 2$, $M = \{m^1, m^2, m^3\}$ with $m^1 = (a_1, b_2, c_1)$, $m^2 = (a_1, b_1, c_2)$ and $m^3 = (a_2, b_2, c_1)$. It can be easily verified that this is a yes-instance.

We now argue that this instance of $1|V_1 \cup V_2, w(V_1) = 0, or\,|\sum w_iC_i$ has a solution with an objective value less than or equal to $\frac{3}{2}q(q+1)$ if and only if the instance of 3DM is a yes-instance. On the one hand, suppose that we have a yes-instance of 3DM; in other words, M contains a matching $M' = \left\{m^1, \ldots, m^q\right\}$ (up to a permutation of indices). We consider the following solution to $1|V_1 \cup V_2, w(V_1) = 0, or\,|\sum w_iC_i$: we first schedule the job m^1 followed by the three successor jobs a^1, b^1, c^1; we proceed with job m^2 and the three successor jobs a^2, b^2, c^2. This schedule continues up to job m^q and the three successor jobs a^q, b^q, c^q. The remaining jobs $m^{q+1}, \ldots, m^{|M|}$ are scheduled afterwards. This solution yields an objective value of $1\times3+2\times3+\ldots+q\times3 = \frac{3}{2}q(q+1)$.

On the other hand, suppose that $T = (t_1, \ldots, t_{3q+|M|})$ is a solution to $1|V_1 \cup V_2, w(V_1) = 0, or\,|\sum w_i C_i$ with an objective value less than or equal to $\frac{3}{2}q(q+1)$. Because of the precedence constraints, at least q jobs in V_1 must be scheduled before the last job in V_2 is scheduled. If $q+1$ jobs in V_1 are scheduled before the last job in V_2 then at least one job in V_2 is scheduled after the $q+1^{\text{th}}$ job in V_1. For that job, the weighted completion time is $1 \times (q+1)$. For the remaining $3q-1$ jobs scheduled before the $q+1^{\text{th}}$ job in V_1, the sum of weighted completion times is at least $1 \times 3 + 2 \times 3 + \ldots + (q-1) \times 3 + q \times 2$. Summing up everything, we have $1 \times 3 + 2 \times 3 + \ldots + q \times 2 + q + 1 = 1 + \frac{3}{2}q(q+1) > \frac{3}{2}q(q+1)$. Therefore, any solution to $1|V_1 \cup V_2, w(V_1) = 0, or\,|\sum w_i C_i$ with an objective value less than or equal to $\frac{3}{2}q(q+1)$ executes exactly q jobs in V_1 before the last job in V_2. Since there are $3q$ jobs in V_2 and each job in V_1 is the predecessor of exactly three jobs in V_2, we infer that the schedule $T = (t_1, \ldots, t_{3q+|M|})$ is such that each of the q first scheduled jobs in V_1 is immediately followed by three jobs in V_2. We consider the subset $M' \subseteq M$ built as follows: a triple m_i belongs to M' if and only if the corresponding job in V_1 is scheduled among the first q such jobs. It is not difficult to see that M' is a matching, which implies that we have a yes-instance of 3DM. This concludes the proof. $\square$

From Lemma 1, we have the following complexity result for Problem 2.

Corollary 2. *Problem 2 is strongly NP-hard, even when $R_\ell = 1$ and $|S_\ell| = 3$ for each group activity ℓ, and $\pi_i = 1/n$ and $t_i = 0$ for each box i.*

In other words, Corollary 2 applies to the case where all group activities have a unit cost and exactly three associated boxes, and all boxes are identical with a zero inspection cost.

5 Some easy subproblems

In this section, we identify special cases of Problem 1 and Problem 2 that can be solved in polynomial time.

5.1 $S_\ell \cap S_{\ell'} = \emptyset$ for any two group activities ℓ and ℓ'

In this case there is no difference between the conjunctive and the disjunctive case, thus Problem 1 and Problem 2 coincide. The precedence graph of the equivalent scheduling problem is a forest of depth two. Therefore, we can solve this special case with Horn's algorithm for a forest [8]. The time complexity is $O(n \log n)$.

5.2 $|S_\ell| = 1$ for each group activity ℓ

For the conjunctive case the group activities can now be eliminated by adding to each box cost t_i the group activity costs R_ℓ for which $i \in S_\ell$. Next we may again apply Smith's rule. Alternatively, since the precedence graph of the equivalent scheduling problem is an upside-down forest (of depth two), we may also apply Horn's algorithm for upside-down forests [8]. For the disjunctive case in each rooted upside-down tree, we keep only one edge (ℓ, i) with smallest cost R_ℓ, reducing the problem to the previous case (Section 5.1). The time complexity for the conjunctive as well as the disjunctive case is again $O(n \log n)$.

5.3 $|S_\ell| = 2$ for each group activity ℓ

We will show that Problem 2 can be solved in polynomial time under the assumotions of Corollary 2 when in addition each group activity has exactly two associated boxes (instead of three). This result is true even if the cost of each group activity is different from one (but the same) and the cost of searching a box is different from zero (but the same). We call a box that does not appear in any subset S_ℓ a *free* box. Furthermore, a solution to an instance of Problem 2 is called a *maximal* solution when it has the form $(\alpha_2, \alpha_1, \alpha_0)$, where each group activity ℓ in α_k is immediately followed by exactly k boxes in S_ℓ $(k = 0, 1, 2)$. We need the following lemma.

Lemma 2. *Any optimal solution to an instance of Problem 2 without free boxes, with $R_\ell = 1$ and $|S_\ell| = 2$ for each group activity ℓ, and with $\pi_i = 1/n$ and $t_i = 0$ for each box i, is a maximal solution.*

Proof. First observe that, since searching a box has a zero cost, it is a dominant decision to search a box as soon as it is available. Therefore we may assume that in any optimal schedule, all unsearched boxes in a set S_ℓ are immediately searched after group activity ℓ is performed. Since each group activity is associated with exactly two boxes, in any optimal solution each group activity is followed by either two, one or zero boxes. Unexecuted group activities ℓ for which all boxes in S_ℓ have already been searched, can be placed at the end of the schedule without increasing the cost. Since we assume that the instance contains no free boxes, it remains to be shown that in an optimal solution we should always search the boxes that can be sequenced in pairs before the boxes that can only be searched one by one. Assume, by contradiction, that there is an optimal schedule for which this is not true. In this schedule we identify the first group activity ℓ that is followed by both boxes in S_ℓ but for which the preceding k boxes are all immediately preceded by a group activity $(k \geq 1)$. If we move ℓ and S_ℓ directly before the group activity that precedes the first of these k boxes, the expected cost of each of the two boxes in S_ℓ decreases with k, whereas the expected cost of each of the k boxes increases by one. This operation thus results in a net decrease of the total expected cost equal to k with $k > 0$, and therefore it cannot be optimal. □

Theorem 3. *Problem 2 is polynomially solvable when each group activity ℓ has equal cost $R_\ell := R$ and $|S_\ell| = 2$, and all boxes i are identical with $\pi_i = 1/n$ and $t_i := t$.*

Proof. Without loss of generality, we may assume that $R = 1$ and $t = 0$. Indeed, if z is the expected cost of any feasible solution when $R = 1$ and $t = 0$, then the expected cost of that same solution for any value of R and t, is $Rz + \frac{1}{n}\sum_{i=1}^{n} it$. We may also assume that there are no free boxes since these can always be scheduled first at zero cost.

Let $\mathcal{I}$ be an instance of Problem 2 that meets all these requirements. In other words, for instance $\mathcal{I}$, each group activity has a unit cost and is associated with exactly two boxes. Furthermore, all boxes are identical with zero inspection cost, and none of the boxes are free boxes. For instance $\mathcal{I}$, we construct an undirected graph G in which each node corresponds with a box, and where there is an edge between two nodes i and j if and only if there is a group activity ℓ such that $S_\ell = \{i, j\}$. The graph G can be constructed in polynomial time. In the remainder of this proof, we will show that an optimal solution to $\mathcal{I}$ can be constructed in polynomial time by finding a maximum-cardinality matching in the graph G.

In this paragraph, we show that with each maximal solution $\alpha = (\alpha_2, \alpha_1, \alpha_0)$ to $\mathcal{I}$, we can associate a maximal matching M in G, and vice versa. On the one hand, for a maximal solution $\alpha = (\alpha_2, \alpha_1, \alpha_0)$, an edge $\{i, j\}$ of G belongs to M if i and j are consecutive boxes in α_2. Edges of M cannot have a node in common because otherwise α_2 would contain a group activity that is immediately followed by only one box. Therefore, M is a matching. The matching M is maximal because α_1 and α_0 do not contain two consecutive boxes. On the other hand, let M be a maximal matching of G with cardinality b. By construction, there are b distinct group activities, say $\ell_1, \dots, \ell_b$, such that $M = \{S_{\ell_1}, \dots, S_{\ell_b}\}$. From this matching M, we can construct a feasible solution α to $\mathcal{I}$ that is maximal, as follows. First, we execute the group activities ℓ_i immediately followed by a search of the two boxes in S_{ℓ_i}, for each $i = 1, \dots, b$. The order in which the group activities are performed or the two boxes in a subset S_ℓ are searched is of no importance because this will not change the expected cost of the solution. Since M is a maximal matching and we have assumed that the instance contains no free boxes, there can be no two boxes of the remaining $n - 2b$ boxes that are searched consecutively. Moreover, for each of those $n - 2b$ unsearched boxes, there exists a different group activity that is not yet scheduled. We may complete the solution α by scheduling each of those group activities, each time followed by a search of the corresponding box. Finally, the remaining group activities are scheduled, if any are left. The construction of α can be done in polynomial time.

Finally, we will prove that a maximal solution to $\mathcal{I}$ is optimal if and only if the associated maximal matching in G is a maximum-cardinality matching. Since the maximum-cardinality matching problem is known to be polynomially solvable [3], the theorem then follows from Lemma 2. The total expected cost of the constructed maximal solution is $\frac{1}{n}(\sum_{i=1}^{b} 2i + \sum_{i=1}^{n-2b}(b+i))$, which equals $\frac{1}{n}(b(b+1) + (n-2b)(n+1)/2)$. By eliminating constant terms and factors, we can see that minimizing this function is equivalent to minimizing $b(b-n)$. The latter function is monotone decreasing in b when $b \leq n/2$. Since the number of edges b in a matching of an undirected graph with n nodes is bounded by $\lfloor n/2 \rfloor$, minimizing $b(b-n)$ is equivalent to maximizing b. $\square$

6 Future work

As future work, we would like to settle the conjunctive case when the number of boxes associated with each group activity is two and three, respectively. If we can show that the latter problem is easy, this would show that, in terms of computational complexity, the disjunctive case is more difficult than the conjunctive case. We would also like to discover the complexity status of the special case of Problem 1 described at the end of Section 3, where the equivalent scheduling problem has a convex bipartite precedence graph. It is worth noting that the instances of Problem 1 for which Wagner and Davis [16] tested their LP model all have this property, and so their experimental findings may suggest that a polynomial algorithm exists in this particular setting. We would also like to see if the result of Corollary 2 remains valid when convexity is added to the problem structure. Finally, we would like to verify whether Theorem 3 can be extended to the case of arbitrary probabilities.

Acknowledgments

Fabrice Talla Nobibon gratefully acknowledges the Fund for Scientific Research - Flanders (FWO-Vlaanderen) for his postdoctoral fellowship. The authors also thank Trivikram Dokka for useful comments.

References

[1] D. Adolphson. Single machine job sequencing with precedence constraints. *SIAM Journal on Computing*, 6:40–54, 1977.

[2] C. Ambühl, M. Mastrolilli, N. Mutsanas, and O. Svensson. On the approximability of single-machine scheduling with precedence constraints. *Mathematics of Operations Research*, 36(4):653–669, 2011.

[3] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[4] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Co., 1979.

[5] D.W. Gillies and J.W.S. Liu. Scheduling tasks with AND/OR precedence constraints. *SIAM Journal on Computing*, 24:797–810, 1995.

[6] B. Gluss. An optimum policy for detecting a fault in a complex system. *Operations Research*, 7:468–477, 1959.

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

[8] W.A. Horn. Single machine job sequencing with treelike precedence ordering and linear delay penalties. *SIAM Journal of Applied Mathematics*, 23(2):189–202, 1972.

[9] W.B. Joyce. Organization of unsuccessful R&D programs. *IEEE Transactions on Engineering Management*, 18(2):57–65, 1971.

[10] J.K. Lenstra and A.H.G. Rinnooy Kan. Complexity of scheduling under precedence constraints. *Operations Research*, 26(1):22–35, 1978.

[11] L.G. Mitten. An analytical solution to the least cost testing sequence problem. *Journal of Industrial Engineering*, 1(11):17–17, 1960.

[12] R.H. Möhring, M. Skutella, and F. Stork. Scheduling with AND/OR precedence constraints. *SIAM Journal on Computing*, 33(2):393–415, 2004.

[13] J.B. Sidney. Decomposition algorithms for single machine scheduling with precedence relations and deferral costs. *Operations Research*, 23(2):283–298, 1975.

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

[15] F. Talla Nobibon, K. Coolen, and R. Leus. A note on 'discrete sequential search with group activities'. *Decision Sciences*, 44(2):395–401, 2013.

[16] B.J. Wagner and D.J. Davis. Discrete sequential search with group activities. *Decision Sciences*, 32(4):557–573, 2001.

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