KATHOLIEKE UNIVERSITEIT
LEUVEN

Faculty of Economics and Applied Economics

An investigation of resource-allocation decisions by means of project networks

Roel Leus

DEPARTMENT OF DECISION SCIENCES AND INFORMATION MANAGEMENT (KBI)

KBI 0626

An investigation of resource-allocation decisions by means of project networks

Roel Leus

Department of Decision Sciences and Information Management, FETEW
Katholieke Universiteit Leuven, Belgium
Roel.Leus@econ.kuleuven.be

This paper investigates the relationship between resource allocation and ES-policies, which are a type of scheduling policies introduced for stochastic scheduling and which can be represented by a directed acyclic graph. We present a formal treatment of resource flows as a representation of resource-allocation decisions, extending the existing literature. A number of complexity results are established, showing that a number of recently proposed objective functions for evaluating the quality of ES-policies lead to difficult problems. Finally, some reflections are provided on possible efficiency enhancements to enumeration algorithms for ES-policies.

Keywords: project scheduling, resource constraints, resource allocation, complexity.

---

1 Introduction

We examine the situation where a set of activities $N = \{0, 1, \ldots, n\}$ is to be scheduled on a number of renewable resource types K with availability a_k, $k = 1, \ldots, K$; N is referred to as a *project*. Activity i has duration $d_i \in \mathbb{N}$ and requires $r_{ik} \in \mathbb{N}$ units of resource type k. Activities 0 and n have zero duration and zero resource usage; we assume that $d_i > 0$ for $i \neq 0, n$. A is a (strict) partial order on N, i.e. an irreflexive and transitive relation, which represents technological precedence constraints. (Dummy) activities 0 and n represent start and end of the project, respectively: $\forall i \in N \backslash \{0\} : (0, i) \in A$, and $\forall i \in N \backslash \{n\} : (i, n) \in A$. We associate the directed acyclic graph (dag) $G(N, A)$ with the partially ordered set (poset) (N, A).

Quantity $s_i \geq 0$ represents the starting time of activity i; starting-time vector $\mathbf{s}$ is a schedule. For an arbitrary relation E on N, define $\mathcal{S}(E) = \{\mathbf{s} \in \mathbb{R}_{\geq}^{n+1} : s_i + d_i \leq s_j, \forall (i, j) \in E\}$, which is a convex polyhedron ($\mathbb{R}_{\geq}$ denotes the set of positive real numbers). $\mathcal{S}(E)$ is non-empty if and only if the

corresponding graph $G(N, E)$ is acyclic. The set of *time-feasible* schedules is $\mathcal{S}(A)$. Clearly, if $A \subseteq E$ then $\mathcal{S}(E) \subseteq \mathcal{S}(A)$.

Without loss of generality, we restrict our attention to integer $\mathbf{s}$. Time interval $[t-1, t]$ is referred to as (time) period t, for integer t. For period $t \in \mathbb{N}_0$, we define $\mathcal{A}(\mathbf{s}, t) = \{i \in N : s_i \leq (t-1) \wedge s_i + d_i \geq t\}$, the activities that are *active* during period t. Schedule $\mathbf{s}$ is called *feasible* if it is time-feasible and *resource-feasible*; the latter property entails

$$\sum_{i \in \mathcal{A}(\mathbf{s},t)} r_{ik} \leq a_k \qquad \forall t \in \mathbb{N}_0, \forall k \in K$$

The resource-constrained project-scheduling problem (RCPSP) $\Gamma(N, A, \mathbf{d}, \mathbf{r}, \mathbf{a})$ aims at finding a feasible schedule that minimizes s_n (see for example [8, 27]), where vectors $\mathbf{d}$, $\mathbf{r}$ and $\mathbf{a}$ collect the activity durations, the resource requirements and the resource availabilities, respectively.

A set of activities $F \subset N$ is a *forbidden set* of an order relation E if it is an antichain of E (a stable set in $G(N, E)$) and if $\sum_{i \in F} r_{ik} > a_k$ for at least one $k \in K$; an inclusion-minimal forbidden set is called a *minimal forbidden set* or *mfs* (see for instance [33]). We denote by $\mathcal{F}(E)$ the set of mfss for order relation E (the parameters of the RCPSP-instance under consideration are also arguments to $\mathcal{F}()$, but are not explicitly mentioned). In the context of scheduling under uncertainty, different scheduling policies for projects with stochastic activity durations were presented in [14] based on the concept of forbidden sets.

For relation E on N, let $T(E)$ denote its transitive closure[1]. A set of policies of interest to us is the set of Earliest-Start policies (ES-policies). The idea is to extend partial order A to $A \cup X$ such that $\mathcal{F}(T(A \cup X)) = \varnothing$, or in other words, such that no $F \in \mathcal{F}(A)$ remains a stable set of $G(N, T(A \cup X))$. The condition for feasibility of the policy is that $G(N, A \cup X)$ still be acyclic. A set $X \subset (N \times N) \backslash A$ of activity pairs that leads to a feasible ES-policy is called a *sufficient* set or selection. $T(A \cup X)$ is an order relation if X is sufficient; this does not necessarily hold for $A \cup X$.

Clearly, if $\mathbf{s} \in \mathcal{S}(A \cup X)$ then $\mathbf{s} \in \mathcal{S}(A)$ and $\mathbf{s}$ is resource-feasible: only antichains of $T(A \cup X)$ can concurrently be active. This is useful when activity durations are variable: the ES-policy defined by X simply computes starting times based on a critical-path recursion. In deterministic scheduling, the Main Representation Theorem of [6] establishes the equivalence between RCPSP (and extensions) and the search for an appropriate extension of A, leading to the so-called *order-theoretic* approach to scheduling [24, 25]. The

[1]The transitive closure of a binary relation E on a set N is the minimal transitive relation E' on N that contains E.

model is an extension of the disjunctive-graph representation of the classical job-shop scheduling problem [29].

This paper is concerned with the determination of desirable ES-policies ('desirable' meaning optimal for a particular objective function, within a particular class). Our contributions are threefold: (1) we present a formal treatment of resource flows, extending the existing literature and indicating links with some papers on network design; (2) we establish a number of complexity results, showing that a number of recently proposed objective functions lead to difficult problems; and (3) we make a number of informal suggestions for efficiency enhancements to enumeration algorithms for ES-policies.

The remainder of this text is organized as follows. In Section 2, we explain how ES-policies are related to so-called resource flows and resource-allocation decisions. The complexity of finding an ES-policy for a number of objective functions is investigated in Section 3, including the case where the attention is restricted to the set of policies that constitute valid resource-allocation decisions for a given schedule. Section 4 contains some suggestions for enhancing current implicit-enumeration procedures for ES-policies. We summarize our findings in Section 5.

2 Definitions and properties

In this section, we present the concept of resource flows (Section 2.1), we indicate some links with a number of articles on network design (Section 2.2), and we examine the relationship between resource flows and ES-policies (Section 2.3).

2.1 Resource flows

The use of *resource flows* has been advanced by various sources, among which [5, 19, 26, 27]; in this article, the word *flow* mostly refers to a resource flow. A flow f assigns to each triple $(i, j, k) \in N \times N \times K$ a value $f(i, j, k) \in \mathbb{N}$. These values must satisfy the following constraints, which constitute flow-conservation constraints and lower and upper bounds on flow through intermediate (non-dummy) nodes:

$$\sum_{j \in N} f(j, i, k) = \sum_{j \in N} f(i, j, k) = r_{ik} \qquad \forall i \in N \backslash \{0, n\}, \forall k \in K \qquad (1)$$

Flow quantity a_k (the availability of resource type $k \in K$) is sent into the network from the dummy start node and collected at the end node:

$$\sum_{j \in N} f(0, j, k) = \sum_{j \in N} f(j, n, k) = a_k \qquad \forall k \in K \qquad (2)$$

Remark that we are not dealing with multi-commodity flow: each resource type has its own distinct capacity. For a flow f we define the set of activity pairs $\phi(f) = \{(i, j) \in N \times N : f(i, j, k) > 0 \text{ for at least one } k \in K\}$, containing the arcs that carry non-zero flow. We also define $C(f) = \phi(f) \backslash A$: the arcs in $C(f)$ are the flow-carrying arcs that do not coincide with technological precedence constraints.

$f(i, j, k)$ represents the number of resource units of type k that are transferred from activity i (when it finishes) to activity j (when it starts). As a result, f entails a detailed resource-allocation decision for the individual resource units that compose a resource type, and induces additional precedence constraints via the elements of $C(f)$. We call flow f *feasible* when $G(N, A \cup C(f))$ is acyclic, in which case the project can be implemented according to the resource-allocation and -routing decisions inherent in f.

We illustrate these definitions by means of an example: consider a project with $N = \{0, 1, 2, 3, 4, 5\}$, so there are four non-dummy project activities. There are no precedence constraints between non-dummy activities: $A = \{(0, 1), (0, 2), (0, 3), (0, 4), (0, 5), (1, 5), (2, 5), (3, 5), (4, 5)\}$. We consider a single resource type ($K = 1$) with availability $a_1 = 4$ and usage by the non-dummy activities $r_1 = r_2 = r_3 = r_4 = 2$ (we write r_{i1} as r_i). A feasible resource flow for this example is given in Figure 1: the four individual resource units are dispatched into the network by dummy node 0 and gathered at node 5.

2.2 Network design

Expressed in very general terms, *network design problems* are concerned with optimally designing a network in order to meet a given set of specifications while minimizing total cost, cfr. [20, 21]. A solution is a network flow, and the cost of a solution is a function of the arc flow values. The related topic of *graph augmentation* studies the problem of augmenting a graph to reach a given requirement (e.g., connectivity) by adding edges [12].

A lot of network design problems are special cases of the (intractable) Minimum Edge-Cost Flow Problem (MECF), problem ND32 in [10], where the goal is to find a maximum flow (obeying edge capacities and flow-conservation laws) so that the cost of edges carrying non-zero flow is minimized. In our

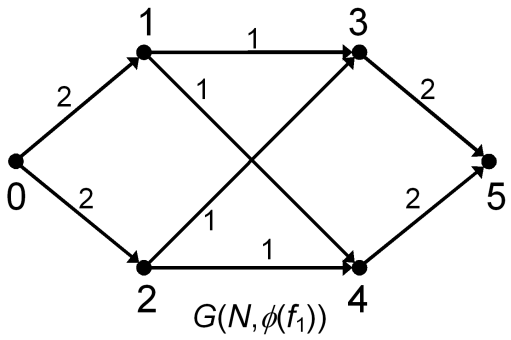

Figure 1: Illustration of a feasible flow f_1 for the example problem. The graphs contains all arcs in $\phi(f_1)$; flow values are indicated on the arcs.

setting, the objective function is indifferent to the *quantity* of flow on each edge, and only distinguishes between zero and non-zero flow (leading to either a fixed charge or a more involved function; see the next sections). The most closely related recent sources we have encountered in the literature on network design are [9, 15, 16, 30]. The major difference with each of these is the fact that we also impose (a) side constraints on *node flow*, namely that it be equal to r_{ik} or a_k (see Equations (1) and (2)), as well as (b) acyclicity.

2.3 The relation between resource flows and ES-policies

We start with two existing results:

Theorem 1. [19]
If f is a feasible flow then $X = C(f)$ is a sufficient set.

For the example presented in Section 2.1, we see that see that $\{(1,3),(1,4),(2,3),(2,4)\}$ is a sufficient set.

Theorem 2. *If X is a sufficient set then a feasible flow f exists with $A \cup C(f) \subseteq T(A \cup X)$.*

Proof: This is shown in [19], based on [22]. □

The following result is stronger than Theorem 1 because the set of selections X to which the statement applies, is enlarged.

Theorem 3. *A selection X is sufficient if $G(N, A \cup X)$ is acyclic and if $\exists$ feasible flow f : $A \cup C(f) \subseteq T(A \cup X)$.*

Proof: The proof presented in [19] can be adapted without fundamental changes if each occurrence of $A \cup X$ is replaced by $T(A \cup X)$. □

The goal of this section is to distinguish a class of sufficient selections that can be useful for algorithmic purposes.

The transitive reduction[2] $t(E)$ of a partial order E is unique [1]. We call a sufficient selection X *dominant* if $(T(A \cup X)\backslash\{(i,j)\})\backslash A$ is *not* sufficient, $\forall (i,j) \in t(A \cup X)\backslash A$. A selection that is not dominant imposes more precedence constraints than necessary and will therefore also be dominated with respect to most 'natural' objective functions (we come back to this at the end of Section 3.2).

The following lemma is useful in what follows:

Lemma 1. *If X is sufficient and $X \subset Y$ and $G(N, A \cup Y)$ is acyclic, then Y is sufficient.*

Proof: This is straightforward: no new antichains can be created by adding pairs to an order relation, and acylicity then leads to sufficiency. □

We can distinguish a condition that is weaker than the one appearing in the definition of dominance:

Lemma 2. *For a sufficient selection X, if X is dominant then $(t(A \cup X)\backslash A)\backslash\{(i,j)\}$ is not sufficient, $\forall (i,j) \in t(A \cup X)\backslash A$.*

Proof: If X is dominant sufficient then $\forall (i,j) \in t(A \cup X)\backslash A$ it holds that $(T(A \cup X)\backslash\{(i,j)\})\backslash A$ is not sufficient. Since $(t(A \cup X)\backslash A)\backslash\{(i,j)\} \subset (T(A \cup X)\backslash\{(i,j)\})\backslash A$ and using Lemma 1, the lemma can be seen to hold. □

Remark that a dominant sufficient selection need not be subset-minimal. This is illustrated in Figure 2 for the example problem that was introduced at the end of Section 2.1: the problem has $\mathcal{F}(A) = \{\{1,2,3\}, \{1,2,4\}, \{1,3,4\}, \{2,3,4\}\}$. For ease of exposition, we omit the dummy start and end node. Selection E_0 in Figure 2 is not sufficient ($\mathcal{F}(T(A \cup E_0)) = \{\{2,3,4\}\}$); selection E_1 is sufficient but not dominant ($t(A \cup E_1)\backslash A = \{(1,2),(1,3),(2,4)\}$, from which $(1,3)$ can be removed while maintaining sufficiency, which allows to invoke (the negation of) Lemma 2); E_2 is sufficient and dominant, but not subset-minimal.

We define an activity pair $(i,j) \in C(f)$ to be *minimal* with respect to feasible flow f if $(i,j) \in t(A \cup C(f))$, or put differently, if apart from arc (i,j),

[2]The transitive reduction of a binary relation E on a set N is the minimal relation E' on N such that $T(E') = T(E)$.

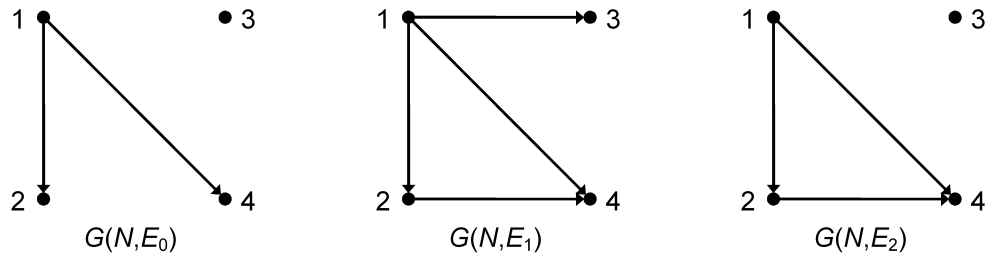

Figure 2: Arc selections for the example problem; dummy nodes 0 and 5 are not included.

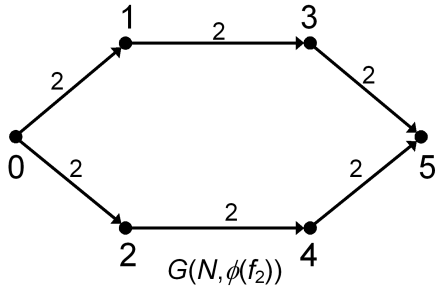

Figure 3: Illustration of a dominant flow f_2 for the example problem.

no path $i \to j$ exists in graph $G(N, A \cup C(f))$ (note that a minimal arc is *not* in A). Feasible flow f is called *dominant* if f has no minimal activity pair (i, j) such that a feasible flow f^* exists with $C(f^*) \subset T(A \cup C(f)) \backslash \{(i, j)\}$. Flow f_1 in Figure 1 is not dominant: minimal arcs $(1, 4)$ and $(2, 3)$ can be removed while the possibility of a feasible flow is preserved; one such flow is f_2 in Figure 3.

Theorem 4. *If X is a dominant sufficient selection then*
(1) a feasible flow f exists with $T(A \cup C(f)) = T(A \cup X)$;
(2) each such feasible flow is dominant;
(3) no feasible flow f exists with $T(A \cup C(f)) \subsetneq T(A \cup X)$.

Proof: Consider a dominant sufficient selection X. By Theorem 2, a feasible flow f exists with $A \cup C(f) \subseteq T(A \cup X)$. From the definition of dominance, we know that $\forall (i, j) \in t(A \cup X) \backslash A$: $(T(A \cup X) \backslash \{i, j\}) \backslash A$ is not sufficient. Consequently, by taking the negation of Theorem 3 and knowing that $G(N, A \cup X)$ is acyclic, no feasible flow f^* exists with $A \cup C(f^*) \subseteq T(A \cup ((T(A \cup X) \backslash \{i, j\}) \backslash A)) = T(A \cup X) \backslash \{i, j\}$. From this, we conclude

that all $(i,j) \in t(A \cup X) \backslash A$ are in $C(f)$ for each feasible flow f. Therefore $T(A \cup C(f)) = T(A \cup X)$, and no feasible flow exists with $T(A \cup C(f)) \subset T(A \cup X)$ for which the inclusion is proper.

Since $A \cup C(f)$ and $A \cup X$ have the same transitive extension, they also have the same transitive reduction. With respect to f, consider any minimal activity pair (i,j); we see that $(i,j) \in t(A \cup X) \backslash A$, and we showed above that no feasible flow f^* exists with $A \cup C(f^*) \subseteq T(A \cup X) \backslash \{i,j\}$. Since $T(A \cup X) = T(A \cup C(f))$, f is dominant. □

Theorem 5. *If f is a dominant feasible flow then $C(f)$ is a dominant sufficient selection.*

Proof: Since f is feasible, $X = C(f)$ is a sufficient set (Theorem 1). Since f is dominant, $\forall (i,j) \in t(A \cup X) \backslash \{(i,j)\}$ it holds that no feasible flow f^* exists with $C(f^*) \subset T(A \cup X) \backslash \{(i,j)\}$. From the negation of Theorem 2, $(T(A \cup X) \backslash \{(i,j)\}) \backslash A$ is not sufficient, so (from the definition of a dominant sufficient selection), X is dominant. □

Lemma 3. *A dominant sufficient selection X is subset-minimal if and only if $X = t(A \cup X) \backslash A$.*

Proof: A sufficient selection X is dominant if $(T(A \cup X) \backslash \{i,j\}) \backslash A$ is not sufficient, $\forall (i,j) \in t(A \cup X) \backslash A$. If $X = t(A \cup X) \backslash A$ then X is subset-minimal because removing any of the elements of X would make X no longer sufficient (by Lemma 2).

Suppose now that X is subset-minimal and that $X \neq X^* = t(A \cup X) \backslash A$. Since $t(A \cup X) \backslash A = t(A \cup X^*) \backslash A$, it follows that X^* also dominant sufficient, and we can see that $X^* \subseteq X$. Therefore $X^* = X$ because strict inclusion would contradict the subset-minimality of X. □

We can now proceed to the most important theorem of this section, where we include the requirement of subset-minimality:

Theorem 6. *A subset-minimal dominant sufficient selection X is exactly the set of minimal arcs of any feasible flow on the network $G(N, T(A \cup X))$.*

Proof: Since X is dominant and sufficient, Theorem 4 says that one or more feasible flows f exist with $T(A \cup C(f)) = T(A \cup X)$, and none with $T(A \cup C(f)) \subsetneq T(A \cup X)$. In the proof of Theorem 4, we showed that all $(i,j) \in t(A \cup X) \backslash A$ are in $C(f)$ for each feasible flow f, so that $t(A \cup X) \backslash A$, which equals X by Lemma 3, are minimal arcs of f. Additionally, $t(A \cup C(f)) = t(A \cup X)$ from which we can conclude that there are no other remaining minimal arcs of f. □

This theorem allows us to partition the set of dominant sufficient selections into equivalence classes with the same subset-minimal representative. The subset-minimal representative of selection E_2 in Figure 2 is $E_3 = \{(1,2),(2,4)\}$; a feasible flow using only arcs from $T(A \cup E_3)$ is f_3 with $\phi(f_3) = \{(0,1),(0,3),(1,2),(2,4),(3,5),(4,5)\}$ and $f_3(i,j) = 2$, $\forall (i,j) \in \phi(f_3)$, where $f_3(i,j,1) \equiv f_3(i,j)$ (in this case, f_3 is also the only feasible flow on the network $G(N, T(A \cup E_3))$).

Finally, we have the following lemma, which says that for each arc (i,j) in a subset-minimal dominant sufficient selection X, it holds that the set $\{i,j\}$ is a subset of at least one mfs that would not be 'resolved' if the corresponding activity pair were removed from X. Here and later, we say that an activity pair $(i,j) \in N \times N$ *resolves* an mfs F if $\{i,j\} \subset F$.

Lemma 4. *For each arc (i,j) in a subset-minimal dominant sufficient selection X, $\exists F \in \mathcal{F}(T(A \cup X) \backslash \{(i,j)\})$ such that $\{i,j\} \subset F$.*

Proof: From Theorem 6, we know that X is exactly the set of minimal arcs of any feasible flow f on the network $G(N, T(A \cup X))$; additionally (Theorem 4(2)), each such f is dominant. Consequently, from the definition of dominant flows, for each $(i,j) \in X$ it holds that no feasible flow f^* exists with $C(f^*) \subset T(A \cup C(f)) \backslash \{(i,j)\}$. Since $T(A \cup X) = T(A \cup C(f))$ (Theorem 4(1)), and from the negation of Theorem 2, we see that $\mathcal{F}(T(A \cup X) \backslash \{(i,j)\})$ contains at least one mfs F (because acyclicity is not an issue here). Since $\mathcal{F}(T(A \cup X)) = \varnothing$, it is the removal of (i,j) that leads to the existence of F. Since $(i,j) \in t(A \cup X)$, we have $T(T(A \cup X) \backslash \{(i,j)\}) = T(A \cup X) \backslash \{(i,j)\}$ and therefore we conclude that $\{i,j\} \subset F$. □

3 The search for optimal policies

In this section, after a brief survey of objective functions encountered in the literature (Section 3.1), we discuss the complexity status of a number of optimization problems (Section 3.2) and we investigate the special case of resource allocation for an input schedule (Section 3.3) as well as the combined problem of resource allocation and scheduling (Section 3.4).

3.1 The objective function

The objective function in stochastic scheduling is usually the expected value of a function of the activity durations, most frequently the project makespan. It turns out that exact determination of such objectives is usually unrealistic [11, 23], and simulation is normally used, cfr. [13, 32]. In [19], the focus

is on the 'stability' objective: the expected value of the deviation between 'baseline' (see Section 3.3) and actual starting times is minimized, weighted and summed over the activities; [19] also resorts to simulation.

Policella [28] proposes to optimize simple functions of the parameters of the constructed dag and thereby implicitly circumvents intractable objective-function evaluation. The first such function is borrowed from [4] and is proportional to the number of incomparable activities in $T(A \cup X)$. In the context of minimizing inter-job communication overhead, minimization of the number of flow-carrying arcs seems to be a valid optimization problem in its own right. Alternatively, Policella suggests that one use a measure of average 'float' in the baseline, relative to the schedule length. Depending on the actual objective function, various float quantities can probably be used (cfr. [34]); [3, 31] suggest the use of free floats, [7] investigates total floats. In Sections 3.2 and 3.3 we examine the construction of ES-policies for the foregoing objectives (based on number of arcs and based on floats), without and with input schedule.

3.2 The complexity of resource allocation

We define the following five related optimization problems:

Problem MinCostFlow
Instance: RCPSP-instance $\Gamma(N, A, \mathbf{d}, \mathbf{r}, \mathbf{a})$, cost coefficients $c_{ij} \in \mathbb{N}$, $\forall (i,j) \in (N \times N) \backslash A$.
Goal: Find a feasible flow f in $G(N, N \times N)$ that minimizes $\sum_{\substack{(i,j) \in C(f) \\ k=1,\ldots,K}} c_{ij} f(i,j,k)$.

Problem MinSuff
Instance: RCPSP-instance $\Gamma(N, A, \mathbf{d}, \mathbf{r}, \mathbf{a})$.
Goal: Find a sufficient set $X \subset N \times N$ with minimum cardinality.

Problem MinFlowArcs
Instance: RCPSP-instance $\Gamma(N, A, \mathbf{d}, \mathbf{r}, \mathbf{a})$.
Goal: Find a feasible flow f in $G(N, N \times N)$ that minimizes $|C(f)|$.

Problem MinMinArcs
Instance: RCPSP-instance $\Gamma(N, A, \mathbf{d}, \mathbf{r}, \mathbf{a})$.
Goal: Find a feasible flow f in $G(N, N \times N)$ with minimal number of minimal arcs.

Problem MaxIncomp
Instance: RCPSP-instance $\Gamma(N, A, \mathbf{d}, \mathbf{r}, \mathbf{a})$.
Goal: Find a sufficient set $X \subset N \times N$ that minimizes $|T(A \cup X)|$.

MinCostFlow is valuable when the cost of extra arcs is proportional to the flow they carry; MinSuff is most interesting from the part of minimal amendment: it yields the smallest number of extra arcs to add to the network; MinFlowArcs and MinMinArcs best reflect the need for limitation of communication overhead; finally, MaxIncomp is useful especially when activity durations are variable: each pair of activities that becomes comparable gives rise to potential starting-time disruptions.

The following result confirms that our definitions of dominance in Section 2.3 make sense:

Theorem 7. *For MaxIncomp, we do not lose all optimal solutions by restriction of the solution space to subset-minimal dominant sufficient selections.*

Proof: A dominant selection X that is not subset-minimal admits the existence of $X^* \subsetneq X$ that is subset-minimal and that has $T(A \cup X) = T(A \cup X^*)$, so that X need not be considered.

Consider a sufficient selection X that is not dominant. Then there exists $(i,j) \in t(A \cup X) \backslash A$ for which $X^* = (T(A \cup X) \backslash \{(i,j)\}) \backslash A$ is sufficient. X need not be considered in the search for optimal selections since $|T(A \cup X^*)| \leq |T(A \cup X)|$. □

Lemma 5. $\min_f |T(A \cup C(f))| = \min_X |T(A \cup X)|$, *where minimization is done over all feasible flows f in $G(N, N \times N)$ and all sufficient sets $X \subset N \times N$, respectively.*

Proof: It follows from Theorem 7 that at least one subset-minimal dominant sufficient selection X minimizes $|T(A \cup X)|$. By Theorem 6, we see that $\min_f |T(A \cup C(f))| \leq \min_X |T(A \cup X)|$.

Likewise, in the search for $\arg\min_f |T(A \cup C(f))|$ we can restrict the search to dominant flows f. It then follows from Theorem 5 that $\min_X |T(A \cup X)| \leq \min_f |T(A \cup C(f))|$. □

In fact, it is not too difficult to show that for any *regular* scheduling objective fuction[3], and using the order-theoretic approach to scheduling, a similar result as Theorem 7 applies. For MinSuff, MinFlowArcs, MinMinArcs and MinCostFlow on the other hand, the same is not true, which will be illustrated below. If an optimization problem is to be used to produce a heuristic solution to a more difficult problem (e.g., to the stochastic RCPSP), this is best kept in mind.

[3]A scheduling objective function is regular if it is non-decreasing in the start times of the activities (for minimization objectives), see e.g. [27].

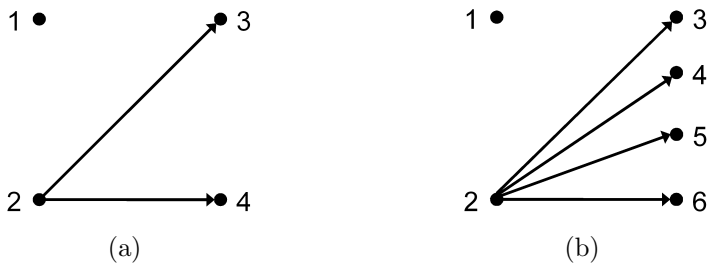

Figure 4: Precedence constraints for the counterexamples; dummy start and end nodes are not included.

Consider an example for which the initial precedence graph is visualized in Figure 4(a). With one resource type ($K = 1$), let $a_1 = r_1 = 2$, $r_3 = r_4 = 1$ and $r_2 = 0$. The mfss are $\mathcal{F}(A) = \{\{1,3\},\{1,4\}\}$, so $T(A \cup X)$ will need to contain at least two extra arcs for a sufficient selection X. If we choose $X^* = \{(1,2)\}$ then $|X^*| = 1$, which is the unique optimal solution for MinSuff. According to our definitions, however, X^* is not dominant, which is logical because $\{1,2\}$ is not even a part of any mfs (remember Lemma 4).

Similarly, dominant flows are not always optimal for MinFlowArcs, MinMinArcs and MinCostFlow. We consider a project with precedence graph $G(N,A)$ as depicted in Figure 4(b). For one resource type ($K = 1$), let $a_1 = 4$ and $r_1 = r_2 = 2$ and $r_3 = r_4 = r_5 = r_6 = 1$. The set of mfss is $\mathcal{F}(A) = \{\{1,3,4,5\},\{1,3,4,6\},\{1,3,5,6\},\{1,4,5,6\}\}$, so $T(A \cup X)$ will again need to contain at least two extra arcs for a sufficient selection X. Multiple feasible flows f^* exist with $C(f^*) = \{(1,2)\}$, which are the only optimal solutions for MinFlowArcs and MinMinArcs. f^* is not dominant, however: again $\{1,2\}$ is not even a part of any mfs. For MinCostFlow, a similar observation can be made for this example if convenient values are chosen for the coefficients c_{ij} (such that they favor f^*).

We now turn our attention to the complexity status of the optimization problems at hand. MinCostFlow is NP-hard, which can be seen by a reduction from HAMILTONIAN PATH (problem GT39 in [10]). To establish the complexity status of the remaining problems, we first define the following decision problem:

Problem X3C (Exact cover by 3-sets)
Instance: A finite set Q with $|Q| = 3q$ and a collection S of 3-element subsets of Q.
Question: Does S contain an *exact cover* for Q, that is, a subcollection $S' \subseteq S$ such that every element of Q occurs in exactly one member of S'?

We know that X3C is NP-complete (see e.g. [10], problem SP2). Without loss of generality, we assume that $|S| \geq q + 2$ and $q \geq 2$.

Theorem 8. *Problem MinFlowArcs is NP-hard.*

Proof: For an arbitrary instance of X3C we construct an RCPSP-instance, as follows. $N = \{0, t\} \cup Q \cup S$, where 0 and t are dummy start and end, respectively (and thereby also predecessor resp. successor of all other activities). $\forall c \in S, u \in Q : (c, u) \in A \Leftrightarrow u \in c$; apart from these pairs, A is empty. There is a single resource type ($K = 1$) with availability $a_1 = |Q|$, with resource usage $r_c = 3$ for each $c \in S$ and $r_u = 1$ for all $u \in Q$. Activity durations are irrelevant here.

Our focus is on minimization of $|C(f)|$. A lower bound on this objective-function value for a feasible flow f is $LB = |S| - q$. This is so because each (non-dummy) activity either obtains all its resource units from 0, or from another predecessor in $G(N, A)$, or it adds at least one unit to the objective function. If we neglect Q then we still have $|S|$ activities with only 0 as predecessor (namely, all activities in S). Since $a_1 = 3q$, activity 0 can supply *all* resource units for at most q of those activities.

Define $\alpha = \{i \in N : f(0, i, 1) > 0\}$; it can be seen that a feasible flow f has $|C(f)| = LB$ only if $\alpha \subset S$, $|\alpha| = q$ and and $\forall i \in \alpha : f(0, i, 1) = 3$. Another necessary condition is that $\forall c \in S \backslash \alpha : |\{i \in N : f(i, c, 1) > 0 \wedge (i, c) \notin A\}| = 1$. Consequently, for each $c \in S \backslash \alpha$ there is an edge $(c_1, c) \in C(f)$ with $c_1 \in S$ and $f(c_1, c, 1) = 3$. This leads to the conclusion that for any feasible flow f for which $|C(f)| = LB$, the set of arcs $Z = \{(i, j) \in T(A \cup C(f)) | i \in \{0\} \cup S \wedge j \in Q \cup \{t\}\}$ is a cut in network $T(A \cup C(f))$ and $\{(i, j) \in T(A \cup C(f)) | (j, i) \in Z\} = \varnothing$. Therefore, the sum of the flow values on the arcs in Z is $3q$ for any feasible f.

If the answer to X3C is 'yes', it is easily seen that a feasible flow f with $|C(f)| = LB$ exists.

Suppose that the answer to X3C is 'no'. In this case, for each feasible flow f, all or part of the resource units used by at least $q + 1$ elements of S are sent to the elements of Q, and/or there is resource flow between elements of Q, and/or $C(f)$ contains arcs (c, u) with $c \in S$ and $u \in Q$. In the second and third case, it is immediate that $|C(f)| > LB$; we further investigate the case where $\nexists (i, u) \in C(f) : u \in Q$. Combining Equations (1) and the foregoing, we know that $\sum_{c \in S} f(c, u, 1) = r_u$ for each $u \in Q$. Since $\sum_{u \in Q} r_u = 3q$, $\nexists c \in S : f(c, t, 1) > 0$. Yet, all or part of the resource units used by at least $q + 1$ elements of S are led to the elements of Q, and so there are at least two activities in S that pass on either one or two resource units, but not three, to another activity in S; we noted higher that this renders equality $|C(f)| = LB$ impossible.

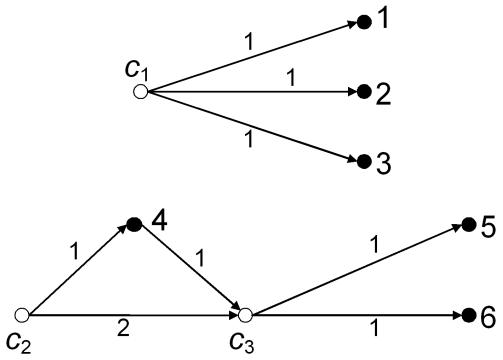

Figure 5: A feasible flow f with number of minimal arcs equal to 1, namely $(4, c_3)$, although $C(f) = \{(c_2, c_3), (4, c_3)\}$. The graph shows all arcs in $\phi(f)$ except the arcs incident to 0 and t; flow values are indicated on the arcs.

This concludes the proof for MinFlowArcs: the answer to an arbitrary X3C-instance is 'yes' if and only if $|C(f)| = LB$ for the corresponding MinFlowArcs-instance, and this transformation can clearly be computed in polynomial time. □

The proof of Theorem 8 can also be used to show intractability of MinMinArcs and MinSuff, but only if we are able to guarantee that non-zero resource flow from $u \in Q$ to $c \in S$ does not occur. We illustrate this with an example: consider an X3C-instance with $q = 2$ and $|S| = 3$ (an extension of the example to the case where $|S| \geq q + 2$ is easily constructed); the elements of Q are indexed from 1 to 6 and $S = \{c_1, c_2, c_3\}$ with $c_1 = \{1, 2, 3\}$, $c_2 = \{1, 2, 4\}$ and $c_3 = \{1, 5, 6\}$; the answer for this instance is 'no' (one reason for this is that object 1 appears in all three sets c_i). Nevertheless, a feasible flow f with number of minimal arcs equal to 1 exists, although $|C(f)| = 2$; one such flow is illustrated in Figure 5. We therefore examine the complexity status of the following slight generalizations:

Problem XMinSuff
Instance: RCPSP-instance $\Gamma(N, A, \mathbf{d}, \mathbf{r}, \mathbf{a})$, set $\nu \subset N \times N$.
Goal: Find a sufficient set $X \subset (N \times N) \backslash \nu$ with minimum cardinality.

Problem XMinMinArcs
Instance: RCPSP-instance $\Gamma(N, A, \mathbf{d}, \mathbf{r}, \mathbf{a})$, set $\nu \subset N \times N$.
Goal: Find a feasible flow f in $G(N, (N \times N) \backslash \nu)$ with minimal number of minimal arcs.

Theorem 9. *Problems XMinMinArcs and XMinSuff are NP-hard.*

Proof: With an arbitrary XC3-instance we associate an RCPSP-instance similar to the one constructed in the proof of Theorem 8. Additionally, set $\nu = \{(u, c) | u \in Q \wedge c \in S\}$.

We first examine XMinMinArcs. Following similar reasoning as before, it can be shown that $LB = |S| - q$ is a lower bound on the number of *minimal* arcs of any feasible flow f, and that if the answer to X3C is 'yes', a feasible flow f exists that achieves this bound.

Suppose that the answer to X3C is 'no': again we only need to investigate the case of feasible f for which $\nexists (i, u) \in C(f) : u \in Q$. The number of minimal arcs is LB only if $\alpha \subset S$, $|\alpha| = q$ and and $\forall i \in \alpha : f(0, i, 1) = 3$, with α defined as before; we investigate an arbitrary such f. Since the answer to X3C is 'no', the resource units used by at least $q + 1$ elements of S are led to the elements of Q, and given the choice for ν, all elements of $S \backslash \alpha$ receive all their resource units from other elements in S. Also, $Z = \{(i, j) \in T(A \cup C(f)) | i \in \{0\} \cup S \wedge j \in Q \cup \{t\}\}$ is a cut in network $T(A \cup C(f))$ and $\{(i, j) \in T(A \cup C(f)) | (j, i) \in Z\} = \varnothing$, and so the sum of the flow values across the arcs in Z is equal to $3q$. Consequently, at least one activity $c_1 \in S \backslash \alpha$ receives resource units from at least two different non-dummy activities $c_2, c_3 \in S$. Either (c_2, c_1) and (c_3, c_1) are both minimal, in which case the total number of minimal arcs exceeds LB, or there exists a path from c_3 to c_2 in $G(N, A \cup C(f))$ (the case with a path from c_2 to c_3 can be treated similarly). In the latter case, c_2 also has two 'resource predecessors' different from 0 (remember that each activity in α receives its resource units *only* from 0), and the same reasoning as for c_1 can be followed. Since $G(N, A \cup C(f))$ is acyclic, we need to repeat this procedure only a finite number of times before we find an activity that has two minimal incoming arcs. This completes the proof for XMinMinArcs.

We now look at XMinSuff, so the minimization of $|X|$ with X a sufficient selection. As a consequence of Theorem 2, and since for each $c \in S$ only $(0, c) \in A$, exactly the same reasoning as for XMinMinArcs can be followed to show the equivalence between 'yes' and 'no' answers to X3C on the one hand, and the minimum cardinality of a sufficient set either equal to or strictly greater than LB, respectively, on the other hand. This completes the proof for XMinSuff. □

Finally, we have the following result for MaxIncomp:

Theorem 10. *Problem MaxIncomp is NP-hard.*

Proof: Lemma 5 stated that minimization of $|T(A \cup X)|$ (by varying X) is equivalent with minimization of $|T(A \cup C(f))|$ (by varying f). We use these two quantities interchangeably in the description that follows, and we will focus on minimization of $obj = |T(A \cup C(f)) \backslash A|$.

For an arbitrary instance of X3C we construct an RCPSP-instance, as follows. N contains $\{0, z, t\} \cup Q \cup S$, where 0 and t are dummy start and end, respectively (and thereby also predecessor resp. successor of all other activities), and z is an extra 'enforcer' activity. $\forall c \in S, u \in Q : (c, u) \in A \Leftrightarrow u \in c$. There is a single resource type ($K = 1$) with availability $a_1 = 3|S|$, with resource usage $r_c = 3$ for each $c \in S$ and $r_u = 1$ for all $u \in Q$; $r_z = 3|S| - |Q|$. We define $LB = |S| - q$ and consider $(3q+1)$ additional sets of LB activities, denoted as sets E_i with $i = 0, 1, \ldots, 3q$, and we construct an arbitrary bijection $\pi : Q \rightarrow \{1, 2, \ldots, 3q\}$. We include each of the sets E_i into N. We also add (e, u) to A, for each $e \in E_0, u \in Q$, as well as (u, e) for each $u \in Q, e \in E_{\pi(u)}$. We set $r_e = 0$ for each $e \in E_i, i = 0, 1, \ldots, 3q$. Although the resulting RCPSP-instance is large, it can be constructed in time polynomial in the size of the X3C-instance.

Any feasible flow f that routes resource units from or to Q-jobs via arcs not in A immediately brings obj above LB. We also see that any $c_1 \in S$ cannot acquire its resource units from another activity $c_2 \in S$ or from z without raising obj above LB. This is true as long as there are no two identical elements in S, and this property can easily be made to hold for the input instance of X3C by removing all but one of such duplicate sets: the 'yes' and 'no' answers remain the same before and after such removal. We also find that any flow f with $obj \leq LB$ necessarily has $f(0, c, 1) = 3$ for each $c \in S$. LB also constitutes a lower bound on obj, since z will obtain its resource units from at least LB different elements of S.

We now claim that the answer to X3C is 'yes' if and only if $obj = LB$. Suppose that the answer to X3C is 'yes': q elements of S can then each be associated with the three elements of Q it contains, and send the resources they use to these Q-activities without cost (because each of these links is already in A). For each of the LB remaining elements c of S, one arc (c, z) can be added, which results in $obj = LB$.

Suppose now that the answer to X3C is 'no': then we will need to add more than LB additional flow-carrying arcs to the network in order to provide z with its resource requirements, and so $obj > LB$.

Consequently, the answer to any X3C-instance is 'yes' if and only if $obj = LB$. This concludes the proof for MaxIncomp. □

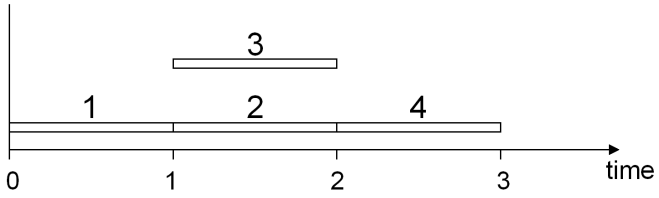

Figure 6: A feasible schedule for the example problem.

3.3 Resource allocation for an input schedule

For a given schedule $\mathbf{s}$ we define the 'schedule-induced' strict order $R(\mathbf{s}) = \{(i,j) \in N \times N | i \neq j \wedge s_i + d_i \leq s_j\}$, which corresponds to the precedence constraints implied by $\mathbf{s}$ [6, 27]. For feasible schedule $\mathbf{s}$, we say that feasible flow f is *compatible* with $\mathbf{s}$ if $\phi(f) \subseteq R(\mathbf{s})$. For instance, flow f_2 in Figure 3 is compatible with the schedule shown in Figure 6 (with $s_0 = 0$ and $s_5 = 3$), but the same is not true for f_1 in Figure 1. A compatible f represents a detailed resource-allocation decision for schedule $\mathbf{s}$. Such resource allocation is useful in case of *static* or *off-line* scheduling, where a 'baseline' schedule for all elements of N is constructed before project execution. In the context of *dynamic* scheduling, the adoption of an ES-policy defined by a resource flow that is compatible with the baseline schedule, constitutes a possibility for dynamic schedule 'repair' in case the actual activity durations are different from those in the baseline. Theorem 11 shows that this latter option is always a feasible alternative.

Theorem 11. [19]
For every feasible schedule $\mathbf{s}$ *exists at least one compatible feasible flow* f*.*

We define the following related optimization problems (the first letter 'S' in each name refers to the fact that a feasible schedule $\mathbf{s}$ is part of the input, as opposed to the problems studied in Section 3.2): SMinCostFlow, SMinFlowArcs and SMinMinArcs, which are the same as their counterparts MinCostFlow, MinFlowArcs and MinMinArcs apart from the fact that we impose $\phi(f) \subset R(\mathbf{s})$, and SMinSuff and SMaxIncomp, which are the same as MinSuff and MaxIncomp apart from the fact that we require $X \subset R(\mathbf{s})$. Remark that the new problems are not sub-problems of their counterparts without schedule since we restrict the set of solutions and not the input parameters.

SMinCostFlow is polynomially solvable: it reduces to a classical network-flow problem; acyclicity is not an issue since $G(N, R(\mathbf{s}))$ is acyclic. For the remaining four problems, we have the following result:

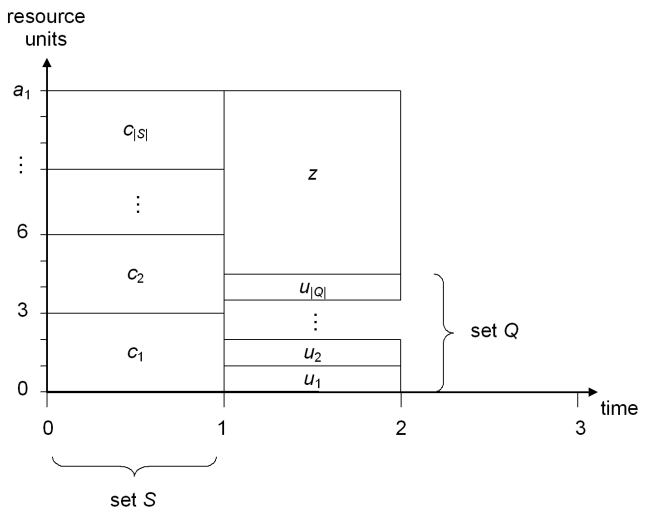

Figure 7: The schedule and resource usage for the proof of Theorem 12.

Theorem 12. *Problems SMinSuff, SMinFlowArcs, SMinMinArcs and SMaxIncomp are NP-hard.*

Proof: For an arbitrary instance of X3C we construct an RCPSP-instance, as follows. $N = \{0, z, t\} \cup Q \cup S$, with z an extra 'enforcer' activity. 0 and t are dummy start and end, respectively. $\forall c \in S, u \in Q$: $(c, u) \in A \Leftrightarrow u \in c$. There is a single resource type ($K = 1$), with resource usage $r_c = 3, c \in S$, $r_u = 1, u \in Q$ and $r_z = 3|S| - |Q|$; availability $a_1 = 3|S|$. All activities in $N \backslash \{0, t\}$ have unit durations. We construct feasible schedule $\mathbf{s}$ for this RCPSP-instance by assigning start times $s_c = 0, c \in S$, $s_z = 1$, $s_u = 1, u \in Q$, and $s_t = 2$. The schedule and resource usage are illustrated in Figure 7.

We first focus on minimization of $|C(f)|$. The reasoning is completely similar to the way in which we associated the value of quantity *obj* to the answer to X3C in the proof of Theorem 10. This concludes the proof for SMinFlowArcs.

Since the arcs used to route the resource units in this proof are all minimal for the resulting resource flow, and since no opportunities exist for lowering the number of minimal arcs, the X3C-instance can also be solved by constructing a flow with minimum number of minimal arcs, so by using an algorithm for solving problem SMinMinArcs as a subroutine.

Additionally, for the described class of SMinMinArcs-instances, the characteristics of *dominance* and of *subset-minimality* are equivalent for a sufficient selection, so the same result would be produced by problem SMinSuff

for the same input data (based on Theorems 5 and 6).

Finally, the theorem is also valid for SMaxIncomp because each of the arcs added in the described solution adds only the minimum possible number of arcs (namely one) to the transitive closure of the resulting extended network. □

Another set of problems of interest to us is based on 'float' or 'slack' values. We define the *pairwise float* between activities i and j to be $\delta_{ij} = s_j - s_i - d_i$, and the *minimum sum of pairwise floats* $\Delta_{ij}(E)$ as the minimal sum of values δ_{kl}, summed over all edges (k, l) on a path from i to j in graph $G(N, E)$, with E an arbitrary order relation on N, minimized over the possibly multiple paths; $\Delta_{ij}(E)$ is defined only for activity pairs $(i, j) \in E$. These definitions lead us to investigate the following problems:

Problem SMaxSumFF
Instance: RCPSP-instance $\Gamma(N, A, \mathbf{d}, \mathbf{r}, \mathbf{a})$, feasible schedule $\mathbf{s}$, integer weights w_i for $i \in N\backslash\{n\}$.
Goal: Find a sufficient set $X \subset R(\mathbf{s})$ that maximizes $\sum_{i \in N\backslash\{n\}} w_i \min_{(i,j) \in T(A \cup X)} \delta_{ij}$.

The allowable increase in the duration of an activity without effect on the starting times of its successors is called 'free float' or 'FF' for short, which explains the choice of the name for this problem.

Problem SMaxSumTF
Instance: RCPSP-instance $\Gamma(N, A, \mathbf{d}, \mathbf{r}, \mathbf{a})$, feasible schedule $\mathbf{s}$, integer weights w_i for $i \in N\backslash\{n\}$.
Goal: Find a sufficient set $X \subset R(\mathbf{s})$ that maximizes $\sum_{i \in N\backslash\{n\}} w_i \Delta_{in}(T(A \cup X))$.

The protection of s_n from delays in the completion of activity i is called 'total float' or 'TF' of i, whence the name of this last problem.

As an illustration, for the schedule in Figure 6 (with $s_5 = 3$) and with sufficient set $C(f_2)$ resulting from flow f_2 as described by Figure 3, we have free-float values of $0, 0, 0, 1, 0$ and total float equal to $1, 1, 0, 1, 0$ for activities $0, 1, 2, 3, 4$, respectively.

In the results below, we will refer to the following decision problem:

Problem 3-PARTITION
Instance: A finite set Q of $3q$ elements, a positive integer B, and an integer size $u(i) > 0$ for each $i \in Q$, such that each $u(i)$ satisfies $B/4 < u(i) < B/2$ and such that $\sum_{i \in Q} u(i) = qB$.
Question: Can Q be partitioned into q disjoint subsets $Q_1, Q_2, \ldots, Q_q$ such that, for $1 \leq i \leq q$, $\sum_{j \in Q_i} u(j) = B$?

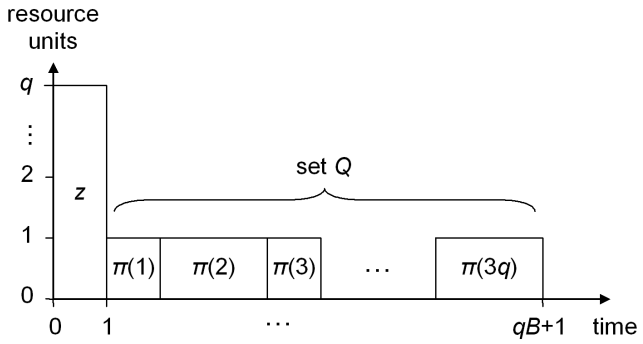

Figure 8: The schedule and resource usage for the proof of Theorem 13.

3-PARTITION is known to be NP-complete (problem SP15 in [10]).

Theorem 13. *Problem SMaxSumTF is NP-hard, even with weights* $w_i \in \{0, 1\}$.

Proof: For an arbitrary instance of 3-PARTITION we construct an instance of RCPSP, as follows. The set of activities to be scheduled is $N = \{0, z, t\} \cup Q$. 0 and t are dummy start and end, respectively. Apart from these dummy precedence constraints, A is empty. There is a single resource type ($K = 1$), with resource usage $r_z = q$ and $r_i = 1, i \in Q$; availability $a_1 = q$. We set $d_z = 1$ and $d_i = u(i)$ for each $i \in Q$. We construct an arbitrary bijection $\pi : \{1, 2, \ldots, 3q\} \rightarrow Q$ and produce feasible schedule $\mathbf{s}$ for this RCPSP-instance by assigning start times $s_z = 0$, $s_{\pi(1)} = 1$ and $s_{\pi(i)} = s_{\pi(i-1)} + d_{\pi(i-1)}$ for $i = 2, 3, \ldots, 3q$; $s_t = qB + 1$. The schedule and resource usage are illustrated in Figure 8. We set weight $w_z = 1$, all other weights are zero.

We claim that the answer to the 3-PARTITION-instance is 'yes' if and only if the optimal objective-function value for the SMaxSumTF-instance is $(q - 1)B$. In that case, each of the q resource units is assigned to (three) elements of Q that sum to exactly B, and the partitioning of Q can follow this resource allocation. Verification of this claim is straightforward. □

Theorem 14. *Problem SMaxSumFF is NP-hard.*

Proof: For an arbitrary instance of 3-PARTITION, we construct an instance of RCPSP, as follows. The set of activities to be scheduled is $N = \{0, t\} \cup Q \cup \{z_1, z_2, \ldots, z_q\}$. 0 and t are dummy start and end, respectively. Apart from these dummy precedence constraints, A is empty. There is a single resource type ($K = 1$), with resource usage $r_i = u(i), i \in Q$, and $r_{z_i} = iB, i = 1, \ldots, q$; availability $a_1 = qB$. All activities in $N \backslash \{0, t\}$ have unit durations. We

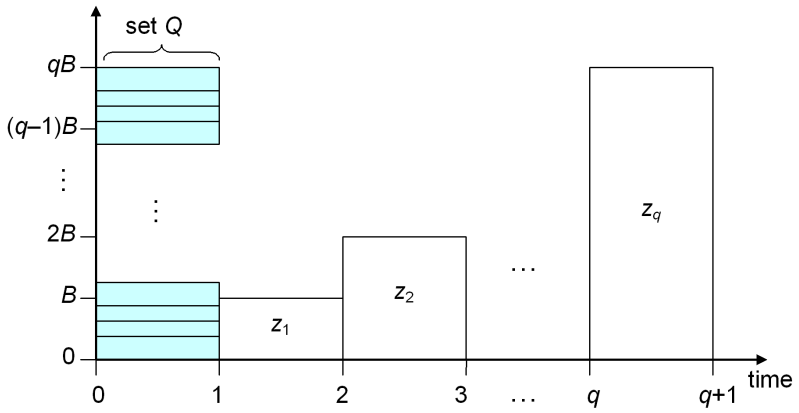

Figure 9: The schedule and resource usage for the proof of Theorem 14.

construct feasible schedule $\mathbf{s}$ for this RCPSP-instance by assigning start times $s_i = 0$ for $i \in Q$, $s_{z_i} = i$ for $i = 1, \ldots, q$ and $s_t = q + 1$. The schedule and resource usage are illustrated in Figure 9.

The weights are selected as follows: $w_0 = 0$, $w_{z_i} = 0$ for $i = 1, \ldots, q$, and $w_i = u(i)$ for $i \in Q$. We assert that the answer to the 3-PARTITION-instance is 'yes' if and only if the optimal objective-function value for the SMaxSumFF-instance is $Bq(q-1)/2$. In that case, set Q can be partitioned into (three-element) subsets with equal sums of values $u(i)$. Verification of the validity of this statement is not difficult. □

This last result is somewhat less satisfactory than Theorem 13, since we allow general values for the weights.

The complexity results of Sections 3.2 and 3.3 confirm that resource allocation is difficult even when objective-function evaluation by itself is not intractable, and indicate that even the use of 'surrogate' objective functions, as has been suggested by a number of recent sources, leads to hard problems, thus justifying the use of solution approaches by means of integer-programming techniques or 'dedicated' implicit enumeration.

3.4 Joint resource allocation and scheduling

Finally, we briefly discuss joint scheduling and resource allocation: this joint problem does not seem to be easier than resource allocation for a given schedule. Actually, this problem deserves study in its own right only if the schedule is an argument to the objective function, which is the case only for the float-based objectives. We define problems MaxSumTF and MaxSumFF (without

initial letter 'S' in the names) similarly as their counterparts SMaxSumTF and SMaxSumFF, but a feasible schedule is now not part of the input but of the output. We also include a deadline D on schedule length, since both total float as well as free float will tend to increase with increasing makespan. We show that both these new problems are difficult:

Theorem 15. *Problems MaxSumFF and MaxSumTF are NP-hard, even with weights* $w_i \in \{0, 1\}$.

Proof: For an arbitrary instance of 3-PARTITION, we construct an instance of RCPSP, as follows. The set of activities to be scheduled is $N = \{0, z, t\} \cup Q$. 0 and t are dummy start and end, respectively. Apart from these dummy precedence constraints, A is empty. Let W equal the sum of the three largest $u(i)$-values in the 3-PARTITION-instance. There is a single resource type ($K = 1$), with resource usage $r_z = q$ and $r_i = 1, i \in Q$; availability $a_1 = q$. Activity z has unit duration, and for $i \in Q$: $d_i = u(i)$. Weight $w_z = 1$, all other weights are zero; deadline $D = W + 1$.

Given our choice of W, a feasible schedule always exists. We claim that the answer to 3-PARTITION is 'yes' if and only if the total float or free float of z is $W - B$. For the objective function (any of the two) to take on this value, the resource profile of the constructed schedule needs to contain a 'gap' in the utilization of each individual resource unit, of length $W - B$. Consequently, activities from Q can only occupy B or less time on each resource unit. If we take into account the duration of the Q-activities, we see that the objective function is $W - B$ only if activities from Q occupy a time period of *exactly* B on each resource unit. Allocation of jobs from Q to resource units therefore corresponds with the desired partitioning of Q. □

We will illustrate the foregoing reduction by means of an example. Consider the following instance of 3-PARTITION: Q contains 12 elements, indexed 1 to 12, with size $u(i) = 3$, $i = 1, \ldots, 8$ and $u(i) = 4$, $i = 9, \ldots, 12$. We have $q = 4$, $B = 10$ and $W = 12$, so we set $a_1 = 4$ and $D = 13$. Figure 10 shows a schedule that achieves total and free float of $W - B = 2$ for z and the answer to the 3-PARTITION-instance therefore is 'yes'; multiple satisfying partitions exist, one being $\{\{1, 5, 9\}, \{2, 6, 10\}, \{3, 7, 11\}, \{4, 8, 12\}\}$, which corresponds with the resource allocation in the figure.

4 Towards more efficient enumeration of solutions?

Existing studies of enumeration (branching) schemes for ES-policies (e.g. [19, 24, 25, 32]) have always dealt with objective functions for which a result

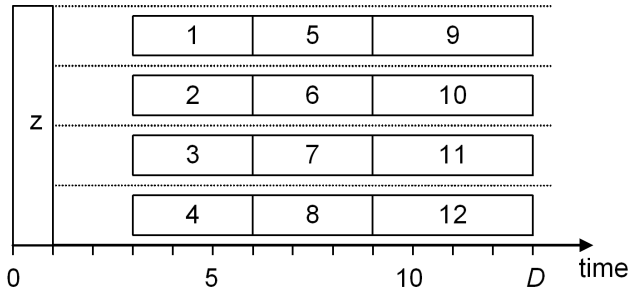

Figure 10: Illustration of the reduction in Theorem 15; the individual resource units correspond with horizontal bands.

similar to Theorem 7 applies: one need only scan the set of subset-minimal dominant sufficient selections. A disadvantage of all current enumeration schemes is that they do not generate only those 'desirable' arc sets: some selections are not dominant, other may be dominant but not subset-minimal. This issue is associated with Lemma 4 together with the fact that answering the question whether a given arc (i, j) resolves any mfs, is NP-complete, which can be shown using similar arguments as [32], where PARTITION is reduced to answering whether or not a given activity $i \in N$ is contained in some (at least one) mfs $F \in \mathcal{F}(A)$. A sufficient selection X need not contain arcs resolving each $F \in \mathcal{F}(A)$, however, and there may exist multiple ways in which to identify a hitting set[4] $X^* \supseteq X$ of arcs in $T(A \cup X)$ such that each mfs is resolved by an element of X^*.

We anticipate that it should be possible to enhance the existing order-theoretic approaches to (both deterministic and stochastic) scheduling based on the concepts introduced in this article. Sequential selection of arcs to extend the input network might follow the approach of [19], which only studies resource allocation for input schedules: a feasible flow f is constructed that is still compatible with all branching decisions (inclusion or exclusion of arcs) made to reach the current position in the search tree, and the arcs in $C(f)$ are one by one included into the selection. The following paragraphs may prove to be useful in the adaptation of this approach so that only dominant selections are generated.

The test whether a given selection is sufficient can be run in polynomial time. For the case where we require that $X \subset R(\mathbf{s})$ (resource allocation for

[4]Consider a collection V of subsets of a set S. A hitting set S' is a subset of S that contains at least one element of each subset in V. For the remark made above, S consists of the arcs in $T(A \cup X)$ and V contains one subset of S for each mfs, with all arcs that resolve the mfs.

input schedule $\mathbf{s}$), acyclicity is not an issue and the existence of a feasible flow is verified by means of maximum-flow computations in a transformed network [19]. In case we demand only that $X \subset N \times N$, such a test can implemented in two steps: (1) test for acyclicity, which either yields a topological ordering of the activities or the observation that the network contains a cycle (see, for instance, [2]); and (2) if a topological ordering is found, the network-flow techniques referred to above can again be invoked.

The reduction of a feasible flow f to a dominant feasible flow f^* can be efficiently accomplished by sending flow across a number of augmenting cycles in the max-flow solution that was referred to supra, in the knowledge that the transitive reduction of an acyclic network can be efficiently determined [1]. Remark that this holds although both problems MaxIncomp and SMaxIncomp are NP-hard.

There is, however, a fundamental difference between resource allocation for an input schedule and the generation of ES-policies without input schedule. This difference resides in the fact that determining the existence of a feasible flow on an oriented network containing cycles, is NP-complete (by reduction from HAMILTONIAN PATH), whereas this is efficiently solvable when the input network is acyclic (based on the network-flow techniques referred to above). This means that for a given schedule, one can efficiently (1) determine a feasible flow that respects the inclusion and exclusion decisions made earlier in the search tree, (2) make this flow dominant, and (3) stepwise include the minimal arcs of the flow into a selection. A branching procedure can then be applied to reverse branching decisions by backtracking, after which step (1) can again be invoked. Without input schedule, however, step (1) already consists of solving a difficult problem, and the suggested procedure becomes considerably less appealing.

It is not the intention of the author to claim that the implementation of the procedures suggested in this section is straightforward; the goal of the foregoing paragraphs was merely to present some informal reflections on possible ways towards more efficient approaches to the generation of optimal solutions. A final isolated but nevertheless related remark to this respect is the observation that the maintenance of transitive closures and transitive reductions of graphs upon edge insertion or deletion has already been the subject of study [17, 18], which may be useful for the development of new implicit-enumeration algorithms.

5 Summary

In this article we have looked into the relationship between resource allocation and ES-policies, which are a type of scheduling policies introduced for stochastic scheduling. We have presented a formal treatment of resource flows as a representation of resource-allocation decisions, extending the existing literature. We have established a number of complexity results, both with and without input schedule, including the cases where the objective is the sum of free floats or the sum of total floats for a given schedule, as well as when the objective is a measure for the extra constraints imposed by the ES-policy on top of the (input) technological precedence constraints; all intractability results were obtained for a single resource type. In a final section, we have also provided some reflections on possible efficiency enhancements to enumeration algorithms for ES-policies.

Our results confirm that resource allocation is difficult even when objective-function evaluation by itself is not intractable, and indicate that even the use of 'surrogate' objective functions, as has been suggested by a number of recent sources, leads to hard problems, thus justifying the use of solution approaches by means of implicit enumeration or integer-programming techniques. Nevertheless, the complexity status of some of the problems that were introduced remains open, e.g. for the problems that were named MinMinArcs and MinSuff.

References

[1] A.V. Aho, M.R. Garey, and J.D. Ullman. The transitive reduction of a directed graph. *SIAM Journal on Computing*, 1:131–137, 1972.

[2] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, 1993.

[3] M.A. Al-Fawzan and M. Haouari. A bi-objective model for robust resource-constrained project scheduling. *International Journal of Production Economics*, 96:175–187, 2005.

[4] M.A. Aloulou and M.-C. Portmann. An efficient proactive reactive scheduling approach to hedge against shop floor disturbances. In G. Kendall, E. Burke, and S. Petrovic, editors, *MISTA 2003. Proceedings of the 1st Multidisciplinary International Conference on Scheduling: Theory and Applications*, volume 1, pages 337–362. Kluwer, 2003.

[5] C. Artigues and F. Roubellat. A polynomial activity insertion algorithm in a multiresource schedule with cumulative constraints and multiple modes. *European Journal of Operational Research*, 127:297–316, 2000.

[6] M. Bartusch, R.H. Möhring, and F.J. Radermacher. Scheduling project networks with resource constraints and time windows. *Annals of Operations Research*, 16:201–240, 1988.

[7] K. Braeckmans, E. Demeulemeester, W. Herroelen, and R. Leus. Proactive resource allocation heuristics for robust project scheduling. Technical Report 0567, Department of Applied Economics, KU Leuven, Belgium, 2005.

[8] E. Demeulemeester and W. Herroelen. *Project Scheduling. A Research Handbook*. Kluwer Academic Publishers, 2002.

[9] G. Even, G. Kortsarz, and W. Slany. On network design: fixed charge flows and the covering steiner problem. *ACM Transactions on Algorithms*, 1:74–101, 2005.

[10] M.R. Garey and D.S. Johnson. *Computers and intractability. A guide to the theory of NP-completeness*. W.H. Freeman and Company, New York, 1979.

[11] J.N. Hagstrom. Computational complexity of PERT problems. *Networks*, 18:139–147, 1988.

[12] T.-S. Hsu. *Graph augmentation and related problems: theory and practice*. PhD thesis, The University of Texas at Austin, Texas, U.S., 1993.

[13] G. Igelmund and F.J. Radermacher. Algorithmic approaches to preselective strategies for stochastic scheduling problems. *Networks*, 13:29–48, 1983.

[14] G. Igelmund and F.J. Radermacher. Preselective strategies for the optimization of stochastic project networks under resource constraints. *Networks*, 13:1–28, 1983.

[15] G. Kortsarz and W. Slany. The minimum shift design problem and its relation to the minimum edge-cost flow problem. Technical Report DBAI-TR-2001-46, DBAI, Institut für Informationssysteme, TU Wien, 2001.

[16] S. Krumke, H. Noltemeier, S. Schwarz, H. Wirth, and R. Ravi. Flow improvement and network flows with fixed costs. In *Proceedings of the international conference on Operations Research (OR-98), Zürich*. Springer, 1998. Available at citeseer.ist.psu.edu/krumke98flow.html.

[17] J.A. La Poutré and J. van Leeuwen. Maintenance of transitive closures and transitive reductions of graphs. Technical Report RUU-CS-87-25, Vakgroep Informatica, Rijksuniversiteit Utrecht, The Netherlands, 1987. Available at http://citeseer.ist.psu.edu/poutre87maintenance.html.

[18] J.A. La Poutré and J. van Leeuwen. Maintenance of transitive closures and transitive reductions of graphs. In H. Göttler and H. J. Schneider, editors, *Proceedings of the 13th International Workshop on "Graph-Theoretic Concepts in Computer Science" (WG '87)*, volume 314 of *Lecture Notes in Computer Science*, pages 106–120. Springer, 1988.

[19] R. Leus and W. Herroelen. Stability and resource allocation in project planning. *IIE Transactions*, 36:667–682, 2004.

[20] T.L. Magnanti and R.T. Wong. Network design and transportation planning: models and algorithms. *Transportation Science*, 18:1–55, 1984.

[21] M. Minoux. Network synthesis and optimum network design problems: models, solution methods and applications. *Networks*, 19:313–360, 1989.

[22] R.H. Möhring. Algorithmic aspects of comparability graphs and interval graphs. In I. Rival, editor, *Graphs and orders*, pages 41–101. Reidel Publishing Company, 1985.

[23] R.H. Möhring. Scheduling under uncertainty: bounding the makespan distribution. In H. Alt, editor, *Computational Discrete Mathematics: advanced lectures*, pages 79–97. Springer-Verlag, New York, 2001.

[24] R.H. Möhring and F.J. Radermacher. The order-theoretic approach to scheduling: the deterministic case. In R. Slowinski and J. Weglarz, editors, *Advances in project scheduling*, chapter I.2. Elsevier, 1989.

[25] R.H. Möhring and F.J. Radermacher. The order-theoretic approach to scheduling: the stochastic case. In R. Slowinski and J. Weglarz, editors, *Advances in project scheduling*, chapter III.4. Elsevier, 1989.

[26] G. Naegler and S. Schoenherr. Resource allocation in a network model - the Leinet system. In R. Slowinski and J. Weglarz, editors, *Advances in project scheduling*, chapter II.8. Elsevier, 1989.

[27] K. Neumann, C. Schwindt, and J. Zimmermann. *Project Scheduling with Time Windows and Scarce Resources: Temporal and Resource-Constrained Project Scheduling with Regular and Nonregular Objective Functions*. Springer-Verlag, 2003.

[28] N. Policella. *Scheduling with uncertainty – a proactive approach using partial order schedules*. PhD thesis, Università degli Studi di Roma "La Sapienza", Rome, Italy, 2005.

[29] B. Roy and B. Sussmann. Les problèmes d'ordonnancement avec contraintes disjonctives. Technical Report Note D.S. no. 9 bis, SEMA, Paris, France, 1964.

[30] S. Schwarz and S.O. Krumke. On budget-constrained flow improvement. *Information Processing Letters*, 66:291–297, 2005.

[31] C. Schwindt. *Resource allocation in project management*. Springer, 2005.

[32] F. Stork. *Stochastic resource-constrained project scheduling*. PhD thesis, TU Berlin, Berlin, Germany, 2001.

[33] F. Stork and M. Uetz. On the generation of circuits and minimal forbidden sets. *Mathematical Programming*, 102:185–203, 2005.

[34] J.D. Wiest and F.K. Levy. *A management guide to PERT/CPM: with GERT/PDM/CPM and other networks*. Prentice-Hall, Inc., Englewood Cliffs, N.J., 1977.