Nash equilibria for the multi-agent project scheduling problem with controllable processing times

Alessandro Agnetis, Cyril Briand, Jean-Charles Billaut, Přemysl Sůcha

Nash equilibria for the multi-agent project scheduling problem with controllable processing times

Alessandro Agnetis[1], Cyril Briand[2,3],
Jean-Charles Billaut[4], Přemysl Šůcha[2,3,5]

[1] Università di Siena ; via Roma 56; 53100 Siena, Italy.
[2] CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France.
[3] Univ de Toulouse, UPS, LAAS, F-31400 Toulouse, France.
[4] Université Francois Rabelais Tours ; Laboratoire d'Informatique ; 64 avenue Jean Portalis, F-37200 Tours, France.
[5] Czech Technical University in Prague; Faculty of Electrical Engineering; Karlovo namesti 13, 121 35, Prague 2, Czech Republic.

Abstract

This paper considers a project scheduling environment in which the activities of the project network are partitioned among a set of agents. Activity durations are controllable, i.e., every agent is allowed to shorten the duration of its activities, incurring a crashing cost. If the project makespan is reduced with respect to its normal value, a reward is offered to the agents and each agent receives a given ratio of the total reward. Agents want to maximize their profit. Assuming a complete knowledge of the agents' parameters and of the activity network, this problem is modeled as a non-cooperative game and Nash equilibria are analyzed. We characterize Nash equilibria in terms of the existence of certain types of cuts on the project network. We show that finding one Nash equilibrium is easy, while finding a Nash strategy that minimizes the project makespan is NP-hard in the strong sense. The particular case where each activity belongs to a different agent

is also studied and some polynomial-time algorithms are proposed for this case.

Keywords. multi-agent project scheduling, Nash equilibria, flow networks.

1 Introduction

A project consists of a set of activities, interconnected by precedence relations, and supposed to reach a particular aim. The project duration, also called *project makespan*, is the minimal time required to perform all the activities. The problem of minimizing the project duration is a basic problem solved by the well known *Critical Path Method*.

In real life contexts, such as building, road trades, aerospace or automotive industry, many large-size projects involve a set of *agents* (actors, firms, organizations, etc.), each being in charge of the execution of a part of the project. The simplest mechanism motivating the agents to complete the project on time is a reward/penalty for project earlier completion/delay.

Multi-agent scheduling problems have been considered in the literature of scheduling, in particular with reference to machine scheduling problems [Agnetis et al., 2004, Leung et al., 2010]. Project scheduling is generally considered with resource constraints and a lot of algorithms are available for solving exactly or heuristically this problem, denoted by RCPSP (Resource Constrained Project Scheduling Problem). In all these views, there is generally one coordinator, centralizing all the decisions.

In the last decade, some papers in the literature consider that the decision making process for a project has many local decision makers, and is by nature decentralized. In [Evaristo and van Fenema, 1999], the authors propose several ways of organization for a project management. Among them, distributed projects where benefits and costs have to be shared are mentioned as interesting research directions. In the seminal work of [Knotts et al., 2000] and [Knotts and Dror, 2003], the authors consider the multi-mode resource constrained project scheduling problems with several agents. Each agent has access to the resources according to a given priority rule (there is one agent per activity). Two types of systems are considered, either with reactive agents or with deliberative agents. The execution mode of the activity is a part of the decision that is let to the agent. Computational experiments are done to test the rules and the different agent systems. The authors in [Brânzei et al., 2002] study penalty division among agents in delayed projects. They propose penalty sharing rules based on cooperative game theory methods. In [Lau et al., 2005a] and [Lau et al., 2005b], the

authors consider the *Distributed Project Scheduling Problem*, where independent and autonomous entreprises must collaborate during the configuration phase and the scheduling phase of the project. These papers describe agent-based approaches, based on negociation between agents (showing the interest of information sharing, possibly partially shared). Following this model, a negotiation-based mechanism is proposed in [Homberger, 2012] and evaluated on some benchmark instances. In [Confessore et al., 2007], the authors consider a resource-constrained "*decentralized multi-project scheduling problem as a multi-agent system, in which multiple agents – corresponding to the local decision makers collaborate, guided by a market-based control mechanism which is coordinated by a coordinator agent.*" An auction mechanism is proposed for agent coordination, as well as heuristic algorithms. In [Wang et al., 2011], the authors consider the *Decentralized Project Scheduling Problem*. A general framework is proposed, considering the resource-constrained version of the problem. Several decision making scenarios are considered (each project being controlled by one project manager, the decision can be associated with an activity, a resource or a coordinator), as well as several organizations (bottom-up, top-down, equivalent). In [Adhau et al., 2012] the authors consider an RCPSP problem where distributed projects share a common set of global resources and they propose a scheduling approach based on a multi-agent system.

In this paper, we do not consider resource constraints. We focus on a scenario in which the agents have their own decisional autonomy and specific competencies and perform specific tasks in a single project. In such a context, project management becomes challenging since each agent naturally pursues the maximization of its own profit, which in turn depends not only on its own decision strategy, but also on the strategies of the other agents and on the satisfaction of the customer. In many practical situations, given a certain quality level for project outcome, customer satisfaction is related to how soon the project is completed, i.e., *project makespan*. In particular, the shorter the project duration, the greater the customer satisfaction. Hence, if the project completes earlier than its original deadline, it is assumed that a *reward* is established for the agents. We refer to [Estévez-Fernández, 2012] for a description of a similar context. Notice that this point of view is equivalent to saying that the customer has contracted a given price for a shorter due date and that financial penalties are applied in case of tardiness, which is more commonly considered in the scheduling literature. Moreover, we consider that the reward sharing mechanism is simple: the total reward is supposed to be divided among the agents according to a ratio *a priori* defined in an agreement.

These ratios may result from exogenous considerations, based on the different roles of the agents in accomplishing project activities, and they can be for instance based on the solution to a related fair division problem [Li and Zhou, 2012, Brânzei et al., 2002]. As a consequence, each agent knows the cost or benefit to him/her deriving from increasing/decreasing the project duration, and each agent's decision can affect everyone else, as long as it affects the project duration.

The scenario that we consider can be viewed as a particular non-cooperative game where players, corresponding to agents, play together for performing a project. Their strategies correspond to the *durations* they choose for carrying out their activities. As in classical project scheduling problems, we suppose that an agent bears a *crashing cost* proportional to the amount an activity duration is reduced, with respect to its normal duration. The project makespan therefore results from all the agents' strategies.

The aim of this paper is to analyze the problem and to characterize *stable* strategies, i.e., in which no agent has an interest in modifying its own strategy, if no other strategy is changed. In this study we concentrate on various complexity issues under the assumption that the project network, as well as the agents parameters, is known. Even under this strong assumption, we show that some problems remain difficult to solve from the computational viewpoint. Moreover, the various concepts and properties analyzed in this paper may be helpful for designing realistic cooperation mechanisms to pursue strategy stability.

Even if we will review some results from PERT/CPM problems (Section 3.2), we wish to emphasize that our analysis is aimed at characterizing Nash equilibria. Its purpose is not to propose an alternative method for solving classical time-cost/trade-off problems.

The paper is organized as follows. In Section 2, the multi-agent project scheduling problem that we consider is formally defined and notations are introduced. In Section 3, we introduce some definitions, illustrated by an example, and some basic notions in the case of a single agent, that will be used in the general case. Section 4 deals with comlexity issues. We show that deciding if a strategy is a Nash equilibrium is in $\mathcal{NP}$, finding a Nash equilibrium can be solved in polynomial time and finding a Nash equilibrium with bounded makespan is NP-complete. Finally, some conclusions and future research directions are presented in Section 5.

2 Problem statement and notations

The *multi-agent project scheduling problem* that we consider is defined by a tuple $< G, \mathcal{A}, \underline{P}, \overline{P}, C, \pi, w >$, where:

- $G = (N, U)$ is an activity-on-arc graph that defines the activity network, where N is the set of nodes ($N = \{0, 1, \ldots, n-1\}$ representing project events), and U is the set of arcs, which is split into subsets U_R and U_D, corresponding to real and dummy project activities, respectively. Nodes 0 and $(n-1)$ represent the project beginning and end, respectively.

- $\mathcal{A} = \{A_1, \ldots, A_m\}$ is the set of m agents. The set of m_u activities assigned to agent A_u is denoted by $\mathcal{T}_u$, and $\mathcal{T}_u \cap \mathcal{T}_v = \emptyset$, $\forall (u, v)$, $u \neq v$, and $\cup_{u=1}^{m}(\mathcal{T}_u) = U_R$. With each activity $(i, j) \in \mathcal{T}_u$ is associated a duration, denoted by $p_{i,j}$, and decided by agent A_u according to its strategy. Without loss of generality, activity durations are assumed to be integer values.

- $\underline{P} = \{\underline{p}_{i,j}\}_{(i,j)\in U}$ and $\overline{P} = \{\overline{p}_{i,j}\}_{(i,j)\in U}$ denote the vectors of *crash* and *normal* durations, respectively. For any activity $(i, j) \in U_R$, we have:

  $$\underline{p}_{i,j} \leq p_{i,j} \leq \overline{p}_{i,j}$$

  For any activity $(i, j) \in U_D$, we have: $\underline{p}_{i,j} = \overline{p}_{i,j} = 0$.

- $C = \{c_{i,j}\}_{(i,j)\in U}$ is the vector of *unit crashing costs*. The cost incurred for shortening the duration of activity (i, j) from $\overline{p}_{i,j}$ to $p_{i,j}$ is equal to $c_{i,j}(\overline{p}_{i,j} - p_{i,j})$ (with $c_{i,j} \geq 0$). We set $c_{i,j} = 0, \forall (i, j) \in U_D$.

- π is a scalar representing the *daily reward* given by the project customer for shortening the total project duration. The *total reward*, given by the project customer for shortening the project duration from $\overline{D}$ to D, is equal to:

  $$\pi(\overline{D} - D).$$

- $w = \{w_u\}_{1\leq u\leq m}$ is a vector specifying how the *reward* is shared by the agents. Agent A_u receives a fraction w_u of the total reward ($w_u \geq 0$, $\forall u$, $1 \leq u \leq m$ and $\sum_{u=1}^{m} w_u = 1$).

A choice for the duration of the activities in $\mathcal{T}_u$ is an *individual strategy* for agent A_u, denoted by P_u. We denote as $\overline{P}_u$ and $\underline{P}_u$ the particular strategies in which $p_{i,j} = \overline{p}_{i,j}$ and $p_{i,j} = \underline{p}_{i,j}$ respectively, $\forall (i, j) \in \mathcal{T}_u$. We have $\underline{P}_u \leq$

$P_u \leq \overline{P}_u$. C_u denotes the vector of crashing costs of activities in $\mathcal{T}_u$. The cost associated to agent A_u for the strategy P_u is:

$$C_u(\overline{P}_u - P_u) = \sum_{(i,j)\in\mathcal{T}_u} c_{i,j}(\overline{p}_{i,j} - p_{i,j}).$$

An overall *strategy profile*, denoted by S, is an m-vector that gathers all individual strategies: $S = (P_1, \ldots, P_m)$. We denote by S_{-u} the $(m-1)$-vector of strategies played by all the agents except agent A_u, i.e., $S_{-u} = (P_1, P_2, \ldots, P_{u-1}, P_{u+1}, \ldots P_m)$. Then, focusing on a particular agent A_u, we have $S = (S_{-u}, P_u)$. We also define $\overline{S} = (\overline{P}_1, \ldots, \overline{P}_m)$ and $\underline{S} = (\underline{P}_1, \ldots, \underline{P}_m)$. For the sake of simplicity, except in case of ambiguity, an overall strategy profile S will be simply referred to as a strategy in the sequel of the text.

Given a strategy S, a path on G having the longest duration is called a *critical path*. This path can be multiple and its length is the *project makespan*, denoted by $D(S)$. All the arcs of a critical path are *critical activities*. It is well known that delaying one of the critical activities increases the overall project makespan. We call *critical graph* the subgraph $\mathcal{G}(S) \subseteq G(S)$ containing only the critical activities. We denote by $\overline{D} = D(\overline{S})$ and $\underline{D} = D(\underline{S})$ the makespan values corresponding to strategies $\overline{S}$ and $\underline{S}$, respectively.

This paper considers a payment scheme where the customer gives a total reward to the agents, if they are able to shorten the overall project makespan, with respect to its maximal value $\overline{D}$. We consider that the reward is specified by *daily reward* π. Given a strategy S, $(\overline{D} - D(S))$ is the makespan reduction and $\pi(\overline{D} - D(S))$ is the total reward associated with strategy S. Each agent A_u receives a fixed ratio w_u of the total reward for one unit of project makespan reduction, i.e., agent A_u receives πw_u. Therefore, for a strategy S, agent A_u receives:

$$w_u\pi(\overline{D} - D(S)).$$

Given a strategy S, the *profit* of agent A_u, denoted by $Z_u(S)$ (or equivalently by $Z_u(S_{-u}, P_u)$) is given by:

$$Z_u(S) = w_u\pi(\overline{D} - D(S)) - C_u(\overline{P}_u - P_u).$$

We denote by $Z(S) = (Z_1(S), \ldots, Z_m(S))$ the *overall profit vector*.

Since every agent aims at maximizing its profit, and assuming without loss of generality that the project starts at time $t_0 = 0$, the multi-agent

project scheduling problem that we consider is a multi-objective optimization problem:

$$\text{Find } S, \quad \text{MAX } (Z_1(S), Z_2(S), \ldots, Z_m(S)) \tag{1}$$

s.t

$$t_j - t_i - p_{i,j} \geq 0, \forall (i,j) \in U \tag{2}$$

$$\underline{p}_{i,j} \leq p_{i,j} \leq \overline{p}_{i,j}, \forall (i,j) \in U \tag{3}$$

$$t_i \geq 0, \forall i \in N, \tag{4}$$

where t_i is the start time of activity $(i,j) \in U$.

Notice that if $\mathcal{A}$ contains a single agent, the problem reduces to the classical time/cost tradeoff project scheduling problem [Phillips and Dessouky, 1977].

3 Definitions and properties

In this section we introduce the main definitions and concepts. Also, we will review some elementary properties from the classical time/cost trade-off problem single-agent case.

3.1 Definitions and example

Definition 1. (*Nash equilibrium*). A strategy $S = (P_1, \ldots, P_m)$ is a *Nash equilibrium* if for all agents A_u and each individual strategy $P'_u \neq P_u$, we have:

$$Z_u(S_{-u}, P_u) \geq Z_u(S_{-u}, P'_u). \tag{5}$$

We further refer to $\mathcal{S}^N$ as the set of Nash equilibria.

Inequality (5) expresses that no agent A_u has interest in changing its strategy from P_u to P'_u, assuming that all other agents keep their own strategies unchanged (remember that Z_u has to be maximized). So, the notion of Nash equilibrium is related to strategy *stability*, which is a desired property in a multi-agent environment.

Definition 2. (*Poor strategy*). A strategy $S = (P_1, \ldots, P_m)$ with project duration $D(S)$ is *a poor strategy* if and only if there exists an agent A_u and an alternative strategy P'_u such that $Z_u(S) < Z_u(S')$ and $D(S') = D(S)$, with $S' = (S_{-u}, P'_u)$.

In other words, S is a poor strategy if and only if there is at least one agent who can individually increase its profit (e.g. by modifying the duration of

some of its activities), without increasing the makespan (and hence without affecting the profit of the other agents). A strategy which is not poor will be called *non-poor*. We denote by $\hat{\mathcal{S}}$ the set of non-poor strategies.

Corollary 1. *Checking whether a strategy is poor can be done in polynomial time.*

Proof. Considering one particular agent A_u and one particular strategy S with makespan $D(S)$, it is easy to derive a linear program (LP) that seeks to test the existence of a strategy S', which maximizes $Z_u(S') - Z_u(S)$, under the constraints that: i) $D(S') = D(S)$ and, ii) only the duration of activities belonging to A_u can change (i.e., $S' = (S_{-u}, P'_u)$). If the optimal objective value of such an LP is strictly greater than 0 then S is obviously a poor strategy. This can be done for every agent. □

Example 1. *Consider the project displayed in Fig. 1, with 5 activities and* $p_{0,1} \in [2,3]$, $p_{0,2} \in [3,5]$, $p_{1,2} \in [1,2]$, $p_{1,3} \in [4,5]$ *and* $p_{2,3} \in [1,3]$, $c_{0,1} = c_{1,2} = c_{1,3} = c_{2,3} = 1$ *and* $c_{0,2} = 2$. *We consider 3 agents* A_1, A_2 *and* A_3 *(their assigned activities are represented with plain, bold and dotted arcs, respectively). The strategy* $S = (2,1,4,3,3)$ *represented in Fig. 2 has makespan* $D(S) = 6$. *This strategy is poor since agent* A_3, *instead of reducing the length of activity* $(0,2)$ *with respect to normal duration, should have better reduced the length of* $(2,3)$, *which has a lower crashing cost. This would lead to a strategy* $S' = (2,1,4,5,1)$. *However, by doing this, activity* $(1,2)$ *is no longer critical, and hence does not need to be shortened. Strategy* $S'' = (2,2,4,5,1)$ *is non-poor.*

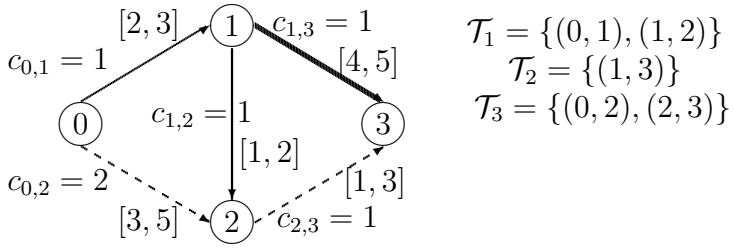

Figure 1: Problem description in Example 1.

Clearly, poor strategies are not interesting since, in particular, a poor strategy cannot be a Nash equilibrium (i.e., $\mathcal{S}^N \subseteq \hat{\mathcal{S}}$).

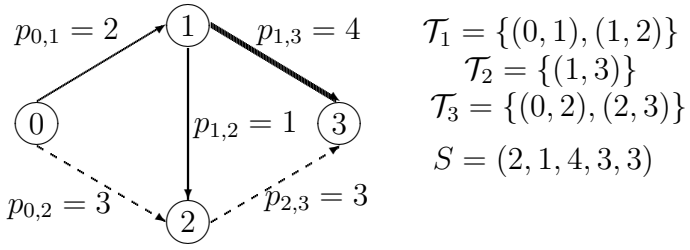

Figure 2: Illustration of a poor strategy.

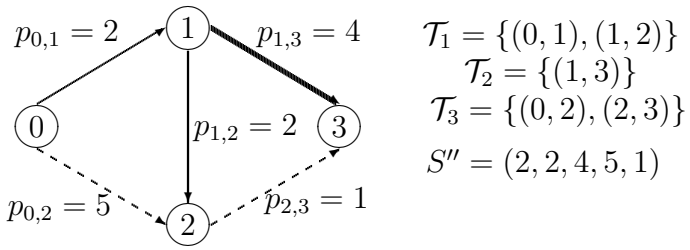

Figure 3: Illustration of a non-poor strategy.

3.2 Project crashing and project extension in the single-agent case

In this section we review a number of classical (hence, single-agent) CPM or PERT concepts.

Notice that if all activities belong to a single agent, a non-poor strategy for a given makespan value is a strategy that minimizes the overall cost.

Given a non-poor strategy S (for a given makespan), a key issue for the agent is to see if there exists a more interesting (non-poor) strategy. Such a strategy can be obtained either by *decreasing the makespan* – which leads to an increase of the total reward despite an increase of the crashing costs, or by *increasing the makespan* – which leads to a reduction of the crashing costs despite a reduction of the total reward.

Given a non-poor strategy S, let $\mathcal{G}(S) = (N^{\mathcal{G}(S)}, U^{\mathcal{G}(S)}) \subseteq G(S)$ be the subgraph containing only the activities that are critical in strategy S. We call it the *critical graph*. Possible modifications to project makespan can be characterized in terms of some specific *cuts* in the critical graph $\mathcal{G}(S)$ [Demeulemeester and Herroelen, 2002].

Definition 3. (*Cut in* $\mathcal{G}(S)$) Given a partition $(X, N \setminus X)$ of the set of activities N such that $0 \in X$ and $n-1 \in N \setminus X$, a *cut* $\omega(X)$ of $\mathcal{G}(S)$ is the

subset of arcs across X and $N \setminus X$. The arcs $(i,j) \in \omega(X)$ with $i \in X$ and $j \in N \setminus X$ are called *forward arcs*, denoted by $\omega^+(X)$. The arcs $(i,j) \in \omega(X)$ with $i \in N \setminus X$ and $j \in X$ are called *backward arcs*, denoted by $\omega^-(X)$. We have $\omega(X) = \omega^+(X) \cup \omega^-(X)$.

We are particularly interested in two types of cuts on $\mathcal{G}(S)$, depending on which action is done on project makespan.

Reduction of project makespan

Given a strategy S and the corresponding critical graph $\mathcal{G}(S)$, we consider the problem of decreasing the makespan of the project at minimum cost. We introduce the definition of a *decreasing cut*.

Definition 4. (*Decreasing cut*) A cut $\omega(X)$ of $\mathcal{G}(S)$ is a *decreasing cut* if

$$\forall (i,j) \in \omega^+(X), p_{i,j} > \underline{p}_{i,j}.$$

The rationale behind this definition is that the makespan of the project can only decrease if the durations of *all* forward arcs of the cut decrease. If an activity is already at crash duration, the cut cannot be used. Therefore, only a *decreasing cut* can be used for decreasing project makespan. In conclusion, the most profitable project makespan reduction is achieved finding the most profitable decreasing cut in $\mathcal{G}(S)$, decreasing its forward activities, and increasing those backward activities that are not at normal duration, if any (so that the obtained strategy is still non-poor). Note that, when using such a cut, the makespan can be reduced gradually until either a new activity not in $\mathcal{G}(S)$ becomes critical in its turn or one of the activities of the cut reaches its minimum value $\underline{p}_{i,j}$.

Increase of project makespan

On the contrary, consider the problem of increasing the makespan. We introduce the definition of an *increasing cut* as follows.

Definition 5. (*Increasing cut)* A cut $\omega(X)$ in $\mathcal{G}(S)$ is an *increasing cut* if:

$$\exists (i,j) \in \omega^+(X) \text{ and } p_{i,j} < \overline{p}_{i,j}$$

$$\forall (i,j) \in \omega^-(X), p_{i,j} > \underline{p}_{i,j}.$$

The rationale behind this definition is the following. Because strategy S is supposed to be non-poor, all the activities that are not in $\mathcal{G}(S)$ are already at their normal durations. Therefore, the only possibility for increasing the makespan is to increase the duration of a critical activity. In conclusion, it must be possible to increase the duration of *at least one* forward activity of the cut, and it must be possible to decrease the duration of *all* backward activities.

For a given S, the cost of activity $(i,j) \in \omega(X)$ related to the unitary change of the makespan can be expressed using two parameters $\ell_{i,j}$ and $u_{i,j}$. Both express unit crashing cost while $\ell_{i,j}$ is considered when $(i,j) \in \omega^+(X)$ and $u_{i,j}$ for $(i,j) \in \omega^-(X)$. The following example illustrates the concept of increasing cut.

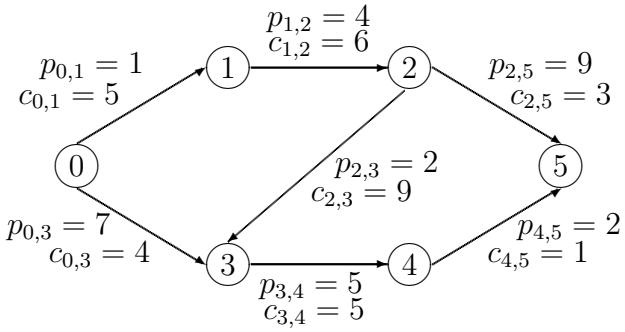

Figure 4: Increasing cuts.

Example 2. *Fig. 4 depicts a strategy S and the corresponding critical graph $\mathcal{G}(S)$, composed by 7 activities. Processing times and crashing costs are indicated in the figure. The makespan is equal to $D(S) = 14$. In order to profitably increase the duration of the project by 1, it would be sufficient to increase the duration of* any *activity by 1, e.g., activity $(1,2)$ (supposing $p_{1,2} < \overline{p}_{1,2}$), which allows to save $\ell_{1,2} = c_{1,2} = 6$. However, since crashing costs are positive, one may observe that, if we increase the duration of $(1,2)$, we can also increase the duration of $(0,3)$ (supposing $p_{0,3} < \overline{p}_{0,3}$), since it generates an additional saving, without further makespan increase. This leads to a unit saving of $\ell_{1,2} + \ell_{0,3} = c_{1,2} + c_{0,3} = 10$. In fact, $(1,2)$ and $(0,3)$ form an increasing cut $\omega(X) = \{(0,3),(1,2)\}$. Notice that if $p_{0,3} = \overline{p}_{0,3}$, $\omega(X)$ is still an increasing cut, but the saving would be given by $c_{1,2} = 6$ only, since $\ell_{0,3} = 0$. Suppose now that one decides to increase $(1,2)$ and $(3,4)$ (instead of $(1,2)$ and $(0,3)$) by 1. In this case, because these latter activities do not form a cut, we observe that the critical*

path $\{(0,1),(1,2),(2,3),(3,4),(4,5)\}$ *would receive a double increase, so the makespan would grow by 2. In fact, in order to increase the makespan exactly by one, the duration of* $(1,2)$ *and* $(3,4)$ *is increased only if activity* $(2,3)$ *can be* decreased*, i.e., we must use the increasing cut* $\omega' = \{(1,2),(3,4),(2,3)\}$ *to get the* saving $\ell_{1,2}+\ell_{3,4}-u_{2,3} = c_{1,2}+c_{3,4}-c_{2,3} = 2$*. Note that if* $p_{2,3} = \underline{p}_{2,3}$*, this would not be possible* ($u_{2,3} = -\infty$)*.*

In the following, we show that, given a strategy S, we only need these two types of cuts to find a more profitable non-poor strategy.

Perturbation of project makespan

Given a non-poor strategy S with makespan $D(S)$, we first consider the problem of finding the most profitable set of activity durations to be increased or decreased, given that the project makespan is minimally perturbed. Since we assume that all values $p_{i,j}$ in S are integer, the minimal makespan perturbation is either $+1$ or -1.

In what follows, for each activity (i,j), let $\alpha_{i,j}$ and $\beta_{i,j}$ denote its *decrease* and, respectively, its *increase* with respect to $p_{i,j}$. Moreover, let the costs $u_{i,j}$ and $\ell_{i,j}$ be defined as indicated in Table 1.

Table 1: Upper and lower capacities of arcs for finding an optimal set of activities to be modified.

Status of the critical arc (i,j)	Values of $\ell_{i,j}$ and $u_{i,j}$
$p_{i,j} = \overline{p}_{i,j}$	$\ell_{i,j} = 0$; $u_{i,j} = c_{i,j}$
$p_{i,j} = \underline{p}_{i,j}$	$\ell_{i,j} = c_{i,j}$; $u_{i,j} = \infty$
$\underline{p}_{i,j} < p_{i,j} < \overline{p}_{i,j}$	$\ell_{i,j} = c_{i,j}$; $u_{i,j} = c_{i,j}$

With these positions, originally devised in [Phillips and Dessouky, 1977], the problem of optimally changing the project makespan can be formulated as a minimum cut problem:

$$\min W(S) = \sum_{(i,j)\in\mathcal{G}(S)} u_{i,j}\alpha_{i,j} - \sum_{(i,j)\in\mathcal{G}(S)} \ell_{i,j}\beta_{i,j} \tag{6}$$

$$\tau_j - \tau_i \geq \beta_{i,j} - \alpha_{i,j} \quad \forall (i,j) \in U^{\mathcal{G}(S)} \tag{7}$$

$$\tau_{n-1} - \tau_0 = \pm 1 \tag{8}$$

$$\alpha_{i,j} \geq 0 \quad \forall (i,j) \in U^{\mathcal{G}(S)} \tag{9}$$

$$\beta_{i,j} \geq 0 \quad \forall (i,j) \in U^{\mathcal{G}(S)} \tag{10}$$

If t_i is the start time of activities outgoing node i under S, here τ_i represents the variation in the value of t_i when project duration is increased by 1 unit (+1 in the rhs of (8)) or, respectively, decreased by 1 unit (-1 in the rhs of (8)). Setting with no loss of generality $\tau_0 = 0$, in the two cases one has, for all i, $\tau_i \in \{0, 1\}$ and, respectively, $\tau_i \in \{-1, 0\}$. The objective function (6) is the total cost from activity crashing, minus the total saving from activity increase. We will denote in the following $W^+(S)$ and $W^-(S)$ the optimal value of the objective function for a unit makespan increase and a unit makespan decrease respectively. Note that one cannot decrease an activity (i, j) having crash duration ($u_{i,j} = +\infty$). Also, setting $\ell_{i,j} = 0$ when an activity (i, j) has normal duration ensures that, in the optimal solution, one has $\beta_{i,j} = 1$ only if (i, j) belongs to an optimal decreasing cut (since there is no convenience to increase an activity which does not bring profit). The meaning of (7) is analogous to that of (2) in (1)–(4), while in (8) we set $\tau_{n-1} = -1$ to decrease the project makespan by 1, and $\tau_{n-1} = 1$ to increase it by 1. Hence, the optimal solution of (6)–(10) specifies a feasible set of duration changes that minimizes overall costs (if $\tau_{n-1} = -1$) or maximizes the savings (if $\tau_{n-1} = 1$). We denote as $(\tau^*, \alpha^*, \beta^*)$ an optimal solution to (6)–(10).

We observe that problem (6)–(10) only makes sense if the optimal solution exists. In fact, consider a strategy S, and suppose that the problem (6)–(10) is unbounded. Then, there exists a feasible solution such that, for a subset $\mathcal{X}$ of activities, it holds $\alpha_{i,j} = \alpha^0_{i,j} + \bar{\alpha}_{i,j}M$ and $\beta_{i,j} = \beta^0_{i,j} + \bar{\beta}_{i,j}M$, for an arbitrarily large M, and

$$\sum_{(i,j)\in\mathcal{X}} u_{i,j}\bar{\alpha}_{i,j} - \sum_{(i,j)\in\mathcal{X}} \ell_{i,j}\bar{\beta}_{i,j} < 0. \tag{11}$$

Moreover, even if some τ_j takes an arbitrarily large value, since $\tau_{n-1} - \tau_0 = \pm 1$, for each path $\mathcal{P}$ from 0 to $n-1$ one has

$$\sum_{(i,j)\in\mathcal{X}\cap\mathcal{P}} \bar{\alpha}_{i,j} - \sum_{(i,j)\in\mathcal{X}\cap\mathcal{P}} \bar{\beta}_{i,j} = 0. \tag{12}$$

Now, consider the strategy S_ϵ obtained by letting $\alpha_{i,j} = \bar{\alpha}_{i,j}\epsilon$ and $\beta_{i,j} = \bar{\beta}_{i,j}\epsilon$ for each $(i, j) \in \mathcal{X}$ for some small $\epsilon > 0$. Due to (11), S_ϵ is strictly better than S, and due to (12), $D(S_\epsilon) = D(S)$. But this implies that S is a poor strategy. Hence, one has:

Theorem 1. *Given a strategy S, if problem (6)–(10) is unbounded, then S is poor.*

In view of Theorem 1, we are ensured that if S is non-poor, problem (6)–(10) has a finite optimal solution. In this case, so does its dual, in which variables x_{ij} are associated with constraints (7) and v with (8):

$$\max \pm v \tag{13}$$

$$\sum_{(h,j)\in\delta^+(j)} x_{h,j} - \sum_{(i,h)\in\delta^-(j)} x_{i,h} = 0 \quad \forall h \in N^{\mathcal{G}(S)} \setminus \{0, n-1\} \tag{14}$$

$$v + \sum_{(0,j)\in\delta^+(0)} x_{0,j} = 0 \tag{15}$$

$$v + \sum_{(i,n-1)\in\delta^-(n-1)} x_{i,n-1} = 0 \tag{16}$$

$$\ell_{i,j} \leq x_{i,j} \leq u_{i,j}. \tag{17}$$

This is a flow problem on a network isomorphic to $\mathcal{G}(S)$, in which arcs have lower and upper capacities $\ell_{i,j}$ and $u_{i,j}$ respectively. Precisely, if we are interested in reducing the makespan at minimum cost, the dual objective function (13) is $+v$, and hence the problem is a max flow problem. If we are interested in increasing the makespan maximizing the saving, then (13) is $-v$, i.e., the problem is indeed to find the *minimum* flow that has to be sent from 0 to $n-1$. Such minimum flow problem can be efficiently solved via either standard max-flow algorithms [Ahuja et al., 1993, p.202], or via flow algorithms which are the symmetrical of classical max-flow algorithms [Ciurea and Ciupalâ, 2004].

Let us illustrate the previous notions with Example 3.

Example 3. *The left side of Fig. 5 shows a project network having 2 activities with* $p_{0,1} \in [1,3]$, $p_{1,2} \in [1,3]$, $c_{0,1} = 1$ *and* $c_{1,2} = 2$. *The strategy* $S_1 = (2,2)$, *illustrated in the middle of the figure, is poor. In fact, the corresponding minimum flow problem is infeasible (i.e., there cannot exist a flow such that* $x_{0,1} = x_{1,2}$ *and* $x_{0,1} \leq 1$ *and* $x_{1,2} \geq 2$*). On the other hand, the minimum flow problem on the right side, for the non-poor strategy* $S_2 = (1,3)$ *having makespan* $D(S_2) = D(S_1) = 4$, *is feasible (*$x_{0,1} = x_{1,2} = 2$*) .*

Since (6)–(10) is the dual of a max- or min-flow problem, it is well known that an optimal solution $(\tau^*, \alpha^*, \beta^*)$ has a very precise structure.

Consider first the case of makespan reduction, i.e., $\tau_{n-1} = -1$. Then, if

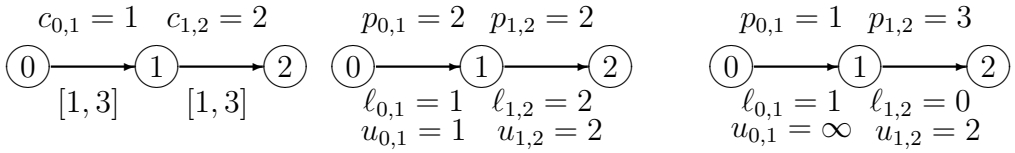

Figure 5: Poorness and flow feasibility.

S is non-poor, there exists a cut $\omega(X^*)$ such that:

$$\begin{aligned}
\tau_i^* &= 0 \text{ if } i \in X^* \\
\tau_i^* &= -1 \text{ if } i \in N \setminus X^* \\
\alpha_{i,j}^* &= 1 \text{ if } (i,j) \in \omega^+(X^*) \\
\beta_{i,j}^* &= 1 \text{ if } (i,j) \in \omega^-(X^*) \\
\alpha_{i,j}^* &= \beta_{i,j}^* = 0 \text{ if } (i,j) \notin \omega(X^*).
\end{aligned}$$

In this case, $\omega(X^*)$ is a decreasing cut. In fact, since $\alpha_{i,j}^* = 1$ for all forward arcs, it means that $p_{i,j} > \underline{p}_{i,j}$ for all of these arcs. Also, note that if $(i,j) \in \omega^-(X^*)$, but (i,j) cannot be extended, then (i,j) becomes non critical after modification.

Suppose now $\tau_{n-1} = 1$ (i.e., we want to increase the makespan maximizing the saving). Then, there exists a cut $\omega(X^*)$ such that:

$$\begin{aligned}
\tau_i^* &= 0 \text{ if } i \in X^* \\
\tau_i^* &= 1 \text{ if } i \in N \setminus X^* \\
\beta_{i,j}^* &= 1 \text{ if } (i,j) \in \omega^+(X^*), p_{i,j} < \overline{p}_{i,j} \\
\beta_{i,j}^* &= 0 \text{ if } (i,j) \in \omega^+(X^*), p_{i,j} = \overline{p}_{i,j} \\
\alpha_{i,j}^* &= 1 \text{ if } (i,j) \in \omega^-(X^*) \\
\alpha_{i,j}^* &= \beta_{i,j}^* = 0 \text{ if } (i,j) \notin \omega(X^*).
\end{aligned}$$

In this case, $\omega(X^*)$ is an increasing cut. In fact, since $\alpha_{i,j}^* = 1$ for all $(i,j) \in \omega^-(X^*)$, it means that $p_{i,j} > \underline{p}_{i,j}$ for all of these arcs. Also, there is at least one $(i,j) \in \omega^+(X^*)$ such that $\beta_{i,j}^* = 1$ (otherwise the makespan cannot be extended). Again, note that if $(i,j) \in \omega^+(X^*)$, but (i,j) cannot be extended, then (i,j) becomes non critical after modification. On the other hand, all backward activities have to be decreased (so all have finite u_{ij}).

In conclusion, given a non-poor strategy S, the most profitable set of activities to be modified is either an optimal decreasing cut with $W^-(S) < \pi$ (the cost of decreasing the makespan is less than the reward) or an optimal increasing cut with $W^+(S) > \pi$ (in this case, the reward loss is less than the saving from crashing cost decrease).

3.3 Multi-agent case

We consider here the multi-agent context. In this case, it is of interest to characterize situations in which a single agent can *individually* decrease or increase the makespan of the project. Hence, definitions 4 and 5 of decreasing and increasing cuts must be suitably modified.

Reduction of project makespan

Let us first consider the problem of makespan *decrease*. An agent A_u can decrease project makespan only if it owns *all* forward activities of a decreasing cut in $\mathcal{G}(S)$ (there is no need to own backward activities).

Definition 6. *A_u-decreasing cut.* A cut $\omega(X)$ in $\mathcal{G}(S)$ is an *A_u-decreasing cut* iff

$$\omega^+(X) \subseteq \mathcal{T}_u$$
$$\forall (i,j) \in \omega^+(X), p_{i,j} > \underline{p}_{i,j}.$$

Given a strategy S and an agent A_u, the problem is to find a subset *of activities of A_u* in $\mathcal{G}(S)$ that have to be modified, in order to optimally decrease the makespan by one time unit. We can use the same approach of Section 3.2 using formulation (6)–(10), but we must now take into account that the duration of activity (i,j) cannot be modified if $(i,j) \notin A_u$. Other agents' activities *can* be backward arcs, but not forward arcs. To this aim, one simply can set $\ell_{i,j} = 0$, $u_{i,j} = \infty$ for these activities (Table 2). We denote by $W_u^-(S)$ the optimal value of the objective function (positive if it is a gain and negative if it is a cost) for agent A_u for a decrease of the project makespan of one time unit.

Increase of project makespan

Let us now consider the problem of makespan *increase*. An agent A_u can individually increase the makespan only if it owns *some* forward activities of an increasing cut in $\mathcal{G}(S)$ and *all* backward activities. In fact, since A_u cannot modify it, a backward activity belonging to another agent would invalidate the computation of the makespan increase.

Definition 7. *A_u-increasing cut.* A cut $\omega(X)$ in $\mathcal{G}(S)$ is *A_u-increasing* iff

$$\omega^+(X) \cap \mathcal{T}_u \neq \emptyset$$
$$\omega^-(X) \subseteq \mathcal{T}_u$$
$$\exists (i,j) \in \omega^+(X) \cap \mathcal{T}_u, p_{i,j} < \overline{p}_{i,j}$$

Table 2: Upper and lower capacities to find an optimal set of activities of agent A_u to be modified.

status of the arc	$\ell_{i,j}$ and $u_{i,j}$
$(i,j) \in A_u$ is critical and $p_{i,j} = \overline{p}_{i,j}$	$\ell_{i,j} = 0$, $u_{i,j} = c_{i,j}$
$(i,j) \in A_u$ is critical and $p_{i,j} = \underline{p}_{i,j}$	$\ell_{i,j} = c_{i,j}$, $u_{i,j} = \infty$
$(i,j) \in A_u$ is critical and $\underline{p}_{i,j} < p_{i,j} < \overline{p}_{i,j}$	$\ell_{i,j} = c_{i,j}$, $u_{i,j} = c_{i,j}$
$(i,j) \notin A_u$	$\ell_{i,j} = 0$, $u_{i,j} = \infty$

$$\forall (i,j) \in \omega^-(X), p_{i,j} > \underline{p}_{i,j}$$

Again, given a strategy S and an agent A_u, one can face the problem of finding the activities of A_u to be modified, in order to optimally increase the makespan by one time unit. We can use the same approach of Section 3.2 using formulation (6)–(10), but we must now take into account that the duration of activity (i,j) cannot be modified if $(i,j) \notin A_u$. Other agents' activities *can* be forward arcs, but not backward arcs. This can still be taken into consideration by setting $\ell_{i,j} = 0$, $u_{i,j} = \infty$ for the activities not belonging to A_u (Table 2). We denote by $W_u^+(S)$ the optimal value of the objective function (positive if it is a gain and negative if it is a cost) for agent A_u for an increase of the makespan by one time unit.

4 Complexity issues

4.1 Deciding if a strategy is a Nash equilibrium

We now want to exploit the concepts presented in the previous section to decide whether or not a given strategy S is a Nash equilibrium. A new element has to be considered, namely the reward $w_u\pi$ that agent A_u earns for each unit of decrease, with respect to $\overline{D}$.

Proposition 1. *If the strategy S is poor, it is not a Nash equilibrium.*

Proof. By definition of a poor strategy. □

Proposition 2. *S is a Nash equilibrium if and only if, for each agent A_u, $u = 1, \ldots, m$:*

$$W_u^+(S) < w_u\pi \tag{18}$$

$$W_u^-(S) \geq w_u\pi. \tag{19}$$

Proof. Consider a strategy S and an agent A_u. If S is non-poor, A_u can improve its situation only by decreasing or increasing the project makespan. In the former case, for a unit reduction of the makespan, A_u receives $w_u\pi$. Hence, such reduction is profitable to A_u if and only if $W_u^-(S) < w_u\pi$. In the latter case, for a unit increase in the makespan, A_u gives up $w_u\pi$, so that it is profitable if and only if $W_u^+(S) \geq w_u\pi$. If one of these conditions holds for one agent, then S is not a Nash equilibrium. Therefore, if and only if conditions (18) and (19) hold for all agents, S is a Nash equilibrium. □

Corollary 2. *Deciding if a strategy is a Nash equilibrium is in $\mathcal{NP}$.*

Proof. The condition given in Proposition 2 can be checked in polynomial time, since, as shown in Section 3.2, $W_u^+(S)$ and $W_u^-(S)$ can be found in polynomial time via max flow algorithms. Therefore we conclude that deciding if a strategy is a Nash equilibrium is in $\mathcal{NP}$.

□

4.2 Finding a Nash equilibrium

Next we prove that finding a Nash equilibrium can be done in polynomial time. The idea is the following.

Let us start from strategy $\overline{S}$ in which all activities have normal duration. Then, agent A_1 determines the duration of its own activities in order to maximize its profit, while the durations of the activities of agents $A_2, \ldots, A_m$ remain unchanged. This can be done by solving a linear program of the type (20) – (23) in which the objective function is Z_1 and $p_{i,j} = \overline{p}_{i,j}$, $\forall (i,j) \notin \mathcal{T}_1$.

$$\text{Find } P_1, \qquad \text{MAX } Z_1(\overline{S}_{-1}, P_1) \tag{20}$$

$$\text{s.t}$$

$$t_j - t_i - p_{i,j} \geq 0, \forall (i,j) \in \mathcal{T}_1 \tag{21}$$

$$\underline{p}_{i,j} \leq p_{i,j} \leq \overline{p}_{i,j}, \forall (i,j) \in \mathcal{T}_1 \tag{22}$$

$$t_i \geq 0, \forall i \in N \tag{23}$$

The returned individual strategy, denoted by P_1^*, is the best response of A_1 to $\overline{P}_{-1} = \{\overline{P}_2, \ldots, \overline{P}_m\}$. Notice that $S_1 = (P_1^*, \overline{P}_2, \ldots, \overline{P}_m)$ is such that $D(S_1) \leq D(\overline{S})$. Notice that the durations of the activities of agent A_1 can only decrease, possibly leading to a makespan decrease.

Next, it is the turn of agent A_2 to compute its optimal individual strategy, denoted by P_2^*, while keeping P_1 fixed to P_1^* and $P_3, \ldots, P_m$ still fixed to $\{\overline{P}_3, \ldots, \overline{P}_m\}$. We obtain strategy $S_2 = (P_1^*, P_2^*, \overline{P}_3, \ldots, \overline{P}_m)$. Again, notice

that the durations of the activities of agent A_2 can only decrease, and hence the makespan can only be unchanged or decreased.

Let consider agent A_1 again. It is possible that:

(i) in $\mathcal{G}(S_2)$, some new critical path appears which was not present in $\mathcal{G}(S_1)$, involving activities of A_1,

(ii) some activities of A_1 which were critical for S_1 are no more critical for S_2 (as it happens to the activities on critical paths that have undergone multiple reductions).

Notice that case (i) does not affect the strategy of agent A_1. In fact, any further makespan reduction due to A_1 would lead to a change of the duration of activities which were already optimally crashed in the previous round. On the other hand, case (ii) can represent an opportunity for A_1 to re-improve its profit, by increasing the duration of some activities that were previously shortened. Therefore, the model (20) – (23) can be used again to re-optimize the strategy of A_1. This case is illustrated in the following example.

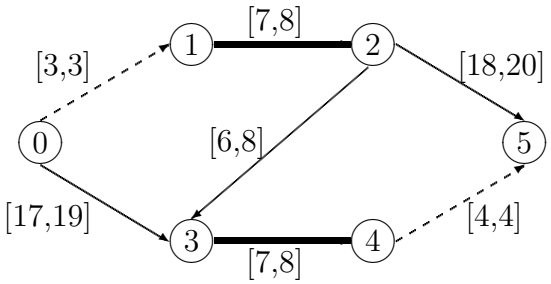

Figure 6: Increasing cuts.

Example 4. *Consider the example of* $\mathcal{G}(\overline{S})$ *in Fig. 6, in which* $\mathcal{T}_1 = \{(0,3), (2,3), (2,5)\}$ *(plain arcs) and* $\mathcal{T}_2 = \{(1,2),(3,4)\}$ *(bold arcs), while activities* $(0,1)$ *and* $(4,5)$ *belong to other agents (dotted arcs). The interval displayed near each arc* (i,j) *is* $[\underline{p}_{i,j}, \overline{p}_{i,j}]$*. The project makespan when activity durations are set to their normal value is* $D(\overline{S}) = 31$ *days, any path from* 0 *to* 5 *being critical. Moreover, suppose that the reward is such that* $w_1\pi > c_{0,3} + c_{2,3} + c_{2,5}$ *and* $w_2\pi > c_{1,2} + c_{3,4}$*. In the first round,* A_1 *can profitably decrease the project makespan by 2 days, crashing all of its activities. The new makespan is 29 and all the activities remain critical. At this point,* A_2 *can crash* $(1,2)$ *and* $(3,4)$ *by 1 day, hence decreasing the makespan to 28. As a consequence, activity* $(2,3)$ *is no more critical and* A_1 *can profitably increase its duration by 1 day, thus saving* $c_{2,3}$*. Notice that increasing the duration of* $(2,3)$ *has*

no effect on A_2, since it does not change the makespan. Therefore, there is no need to re-optimize the strategy of A_2.

This line of reasoning leads to Algorithm 1 called FindNash. This algorithm uses the routine Opt$(Z_k|\beta)$ based on the model (20) – (23), to compute the individual strategy P_k that maximizes Z_k (profit of agent A_k), for fixed strategies of other agents (specified in the field β).

Algorithm 1 FindNash

$(P_1, P_2, \ldots, P_m) \leftarrow (\overline{P}_1, \overline{P}_2, \ldots, \overline{P}_m)$
for $u = 1$ to m **do**
  Compute $P_u^* =$ Opt$(Z_u|P_1, \ldots, P_{u-1}, P_{u+1}, \ldots, P_m)$
  $(P_1, P_2, \ldots, P_u, \ldots, P_m) \leftarrow (P_1, P_2, \ldots, P_u^*, \ldots, P_m)$
  for $v = 1$ to $u - 1$ **do**
    // readjustment phase
    $P_v^* \leftarrow$ Update of P_v by increasing previously crashed, non-critical activities so that they reach their normal duration or become critical.
    $(P_1, P_2, \ldots, P_v, \ldots, P_m) \leftarrow (P_1, P_2, \ldots, P_v^*, \ldots, P_m)$
  end for
end for
return $S^N = (P_1, P_2, \ldots, P_m)$

The agents are considered in the increasing order of their index. When Opt is called with objective Z_u, strategies $P_1^*, P_2^*, \ldots, P_{u-1}^*$ have already been computed, while the activities of agents $A_{u+1}, \ldots, A_m$ are still at normal duration. Such call produces a strategy P_u^*, which is therefore the currently best outcome A_u can reach by its own. Notice that in this phase, the project makespan can be reduced thanks to A_u (it cannot increase since so far all activities of A_u were at normal duration).

After Opt is run on Z_u, some activities of the first $u - 1$ agents may have become non-critical. Therefore, if these activities have been shortened in a previous stage, their duration can be increased, bringing some profit to the agents who own them, and hence updating the corresponding strategies. This can be done by simply increasing the duration of these activities, until either the activity reaches normal duration or becomes critical again. Note that such a readjustment does not affect the makespan, and hence the other agents.

We want to point out that even though the above procedure indicates how each agent can act individually to reach a stable strategy profile, it does not pretend to actually mimic the agents' behavior in a real setting. Indeed, in a real context, in general agents will not share their information, unless

they agree to do so. Here we only intend to show that, assuming complete knowledge, a Nash equilibrium can be found in polynomial time.

Proposition 3. *A Nash strategy can be found in polynomial time.*

Proof. Consider the strategy S^N returned by algorithm FindNash. First, it is obviously a non-poor strategy since the readjustment phase (Alg. 1 lines 7–8) ensures that no agent is able to make more profit without changing the makespan. Moreover, in the critical graph $\mathcal{G}(S^N)$ there cannot be any A_u-increasing cut, since the forward activities of such cut would not have been shortened during the call to Opt on Z_u. Similarly, an A_u-decreasing cut in $\mathcal{G}$ would contain a cut that existed when Opt was called on Z_u. If it was not profitable for Opt to use this cut, it cannot be profitable now. Then, from Property 2, S^N is a Nash equilibrium.

Since it focuses on a single agent profit, Opt can be implemented in polynomial time, using linear programming for instance. The readjustment phase can be done in linear time (in the number of activities), since each activity can be expanded independently from the others. In conclusion, Algorithm FindNash runs in polynomial time since it involves the solution of $\frac{1}{2}m(m+1)$ linear programs. □

Let us note that since algorithm FindNash always yields a strategy, there always exists at least one Nash strategy.

Notice also that the quality of a Nash equilibrium can be arbitrarily bad, as illustrated by Example 5.

Example 5. *In the example illustrated by Fig. 7, there are two agents and one activity per agent with durations in* $[1, 1000]$*. Each activity has a crashing cost of 1, and a reward of 2 is obtained by each agent in case of makespan decrease of 1 time unit. It is clear that the interest of each agent is to fix the duration to 1 time unit. However, strategy* $S^N = (1000, 1000)$ *is a Nash equilibrium (obtained by Algorithm* FindNash*).*

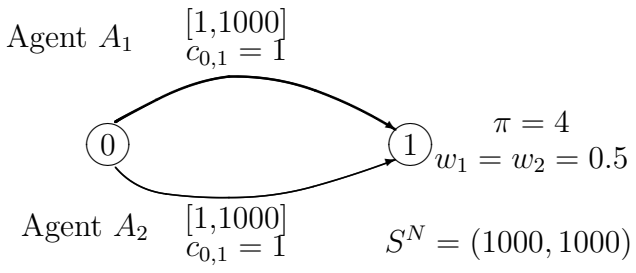

Figure 7: Nash equilibrium of bad quality

Next, we consider the problem of finding a Nash equilibrium with bounded makespan.

4.3 Finding a Nash equilibrium with bounded makespan

We consider the problem of determining whether a strategy which is a Nash equilibrium, and having a makespan not greater than a given value, exists. Indeed, the stability of a strategy is important since it ensures that agents can trust each other, none of them having incentive to modify its own strategy. Moreover, the project customer is certainly interested in determining the minimum makespan that can be reached, provided the organization remains stable.

The decision version of the problem can be written as follows.

NE-BoundedMakespan

Instance:	a tuple $< G, \mathcal{A}, \underline{P}, \overline{P}, C, \pi, w >$ as defined in Section 2, an integer λ.
Question:	Is it possible to find a strategy S such that S is a Nash equilibrium and $D(S) \leq \lambda$?

Proposition 4. *Problem NE-BoundedMakespan is strongly NP-complete.*

Proof. Given a strategy S, it can be checked in polynomial time if it is a Nash equilibrium (see Section 4.1). Furthermore, it can be checked in polynomial time whether or not the makespan exceeds λ. Therefore, the problem belongs to $\mathcal{NP}$.

In order to prove the NP-completeness, we show that 3-Partition problem [Garey and Johnson, 1979] reduces to NE-BoundedMakespan.

3-PARTITION

Instance:	a set $\zeta = \{a_1, \ldots, a_K\}$ of $K = 3k$ positive integers, such that $\sum_{j=1}^{3k} a_j = kB$ and $B/4 < a_j < B/2$ for each j.
Question:	Can ζ be partitioned into k subsets $\zeta_1, \zeta_2, \ldots, \zeta_k$ so that the sum of the integers in each subset is equal to B?

Consider an arbitrary instance of 3-Partition. We construct an instance of the multi-agent project scheduling problem as follows. The activity-on-arc graph G has $3k$ parallel paths, each consisting of k activities, plus two dummy nodes 0 and $3k(k+1)+1$ (see Fig. 8). The activities on the first path are $(1,2),(2,3),\ldots,(k,k+1)$, those on the second path are $(k+2,k+3),(k+3,k+4),\ldots,(2k+1,2k+2)$, and so on, the last path containing activities $(3k(k+1)-k, 3k(k+1)-k+1), (3k(k+1)-k+1, 3k(k+1)-k+2), \ldots, (3k(k+1)-1, 3k(k+1))$. Vertices 0 and $3k(k+1)+1$ represent the start time and

the finish time of the project respectively, and there are dummy activities from the start node $(0,1),(0,k+2),\ldots$ and dummy activities to the ending node $(k+1,3k(k+1)+1),(2k+2,3k(k+1)+1),\ldots$

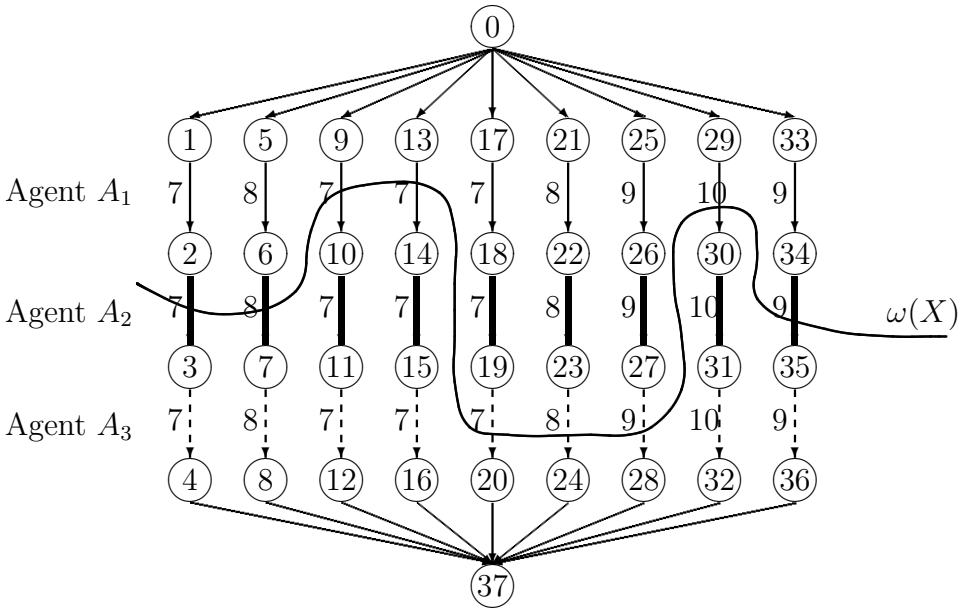

Figure 8: Reduction from 3-PARTITION with $k = 3$.

There are k agents. Agent A_u owns the u^{th} activity of each path, $u = 1,\ldots,m$. All real (non-dummy) activities have $\underline{p}_{i,j} = 0$ and $\overline{p}_{i,j} = 1$. Also, the crashing cost of each activity on the i-th path is a_i, $i = 1,\ldots,3k$. Each agent's unit reward is $w_u\pi = B + \varepsilon$, identical for all agents, ε being an arbitrarily small positive value. We aim at determining whether it exists a Nash strategy S such that $D(S) < k$.

Consider the strategy $\overline{S}$ where all activities have normal duration, $p_{i,j} = 1$. The project makespan for $\overline{S}$ is clearly $D(\overline{S}) = k$. With respect to $\overline{S}$, we observe that an agent may decrease the makespan by $\delta \in (0,1]$, crashing all its activities by the same amount δ. However, by doing this, the agent would pay $kB\delta$ and would only gain $(B+\varepsilon)\delta$. Hence, the new strategy would not be a Nash equilibrium, since the agent would rather get back to $\overline{S}$. In order to obtain a Nash equilibrium, for each agent the total unit crashing cost of the activities involved must not exceed B. As a consequence, no more than three activities per agent can be involved in the decrease. Due to the topology of the graph, in order to decrease the makespan, *exactly* $3k$ activities must be crashed, and therefore, for a Nash equilibrium with makespan smaller than k

to exist, there must be a cut ω containing exactly three activities per agent, inducing an A_u-decreasing cut for each agent, i.e., ω must have exactly one activity per path. The total crashing cost of such a set of activities is kB, so in order to get a Nash equilibrium, the total cost of the three activities selected for each agent must be exactly B. Clearly, this is possible if and only if a partition in the 3-PARTITION instance exists. □

Fig.8 illustrates the construction of the project network from a 3-PARTITION instance characterized by $k = 3$, $\zeta = \{7, 8, 7, 7, 7, 8, 9, 10, 9\}$ and $B = 24$. The problem is to determine, if it exists, a Nash strategy such that the project makespan is less than $k = 3$. If the activities of the cut ω are decreased by $\delta = 1$, we get a Nash equilibrium, which shows that the original instance of 3-PARTITION was a yes-instance.

4.4 The special case $|\mathcal{T}_u| = 1$, $\forall u$

In this section, we consider the special case where each agent owns exactly one activity, so in the sequel we use u to indicate the activity of agent A_u. For this special case, we show that finding a minimum-makespan Nash equilibrium is easy.

By definition, a unit time of makespan decrease brings A_u the reward $w_u\pi$. Since A_u only manages a single activity u, this quantity can be easily compared with the unit crashing cost c_u of u. The set $\mathcal{A}$ of agents can be partitioned into two subsets $\mathcal{A}^+$ and $\mathcal{A}^-$ as follows.

$$\mathcal{A}^+ = \{A_u, 1 \leq u \leq m | c_u < w_u\pi\} \tag{24}$$

$$\mathcal{A}^- = \{A_u, 1 \leq u \leq m | c_u \geq w_u\pi\} \tag{25}$$

For an agent $A_u \in \mathcal{A}^+$, it is profitable to crash its activity if it also reduces the project duration (the profit $w_u\pi - c_u > 0$), while if $A_u \in \mathcal{A}^-$, it is systematically not interesting for A_u to crash the duration of its activity.

Now, consider the initial strategy $\hat{S} = (\hat{P}_1, \ldots, \hat{P}_m)$ defined by

$$\hat{P}_u = \underline{p}_u, \quad A_u \in \mathcal{A}^+ \tag{26}$$

$$\hat{P}_u = \overline{p}_u, \quad A_u \in \mathcal{A}^-. \tag{27}$$

Note that $\hat{S}$ can be a poor strategy, since the activities of $\mathcal{A}^+$ have crash duration even if they do not belong to the critical path $\mathcal{G}(\hat{S})$. Therefore, we can consider these activities one by one (in an arbitrary order), increasing their durations until either the activity becomes critical or reaches its normal duration. By doing so, a non-poor strategy $\tilde{S}$ is obtained, with $D(\tilde{S}) = D(\hat{S})$. Notice that, since at least one path in $\mathcal{G}(\tilde{S})$ was present in $\mathcal{G}(\hat{S})$ as well, the following property holds.

Proposition 5. *In $\mathcal{G}(\tilde{S})$, there is at least one path $\mathcal{P}$ (therefore critical) such that, $\forall u \in \mathcal{P}$:*

- *if $A_u \in \mathcal{A}^+$, $p_u = \underline{p}_u$*
- *if $A_u \in \mathcal{A}^-$, $p_u = \overline{p}_u$.*

Notice that $\tilde{S}$ may not be unique, since different non-poor strategies can be reached, depending on the order in which noncritical activities in $\mathcal{A}^+$ are processed.

Theorem 2. *Strategy $\tilde{S}$ is a Nash equilibrium, and there is no Nash equilibrium with a smaller makespan.*

Proof. Let us first prove that $\tilde{S}$ is a Nash equilibrium. Consider the critical paths on $\mathcal{G}(\tilde{S})$, and let consider $A_u \in \mathcal{A}^-$. Since the activities of $\mathcal{A}^-$ are at their crash duration in $\mathcal{G}(\tilde{S})$, they can improve their situation only through an A_u-decreasing cut. Such a cut exists if it contains only u, and hence the value of the cut is c_u. Since $A_u \in \mathcal{A}^-$, $c_u > w_u\pi$, so no agent $A_u \in \mathcal{A}^-$ has convenience in shortening its activity.

Now let consider $A_u \in \mathcal{A}^+$. If $p_u = \underline{p}_u$ in $\tilde{S}$, A_u can improve its situation only through an A_u-increasing cut. However, in this case $c_u \leq w_u\pi$, so there is no interest for A_u in increasing the makespan. If $p_u > \underline{p}_u$, A_u might search for an A_u-decreasing cut. However, a cut can be A_u-decreasing only if all its forward arcs belong to A_u, and this is not possible since, due to Proposition 5, there is at least another critical path which does not include u.

In conclusion, the conditions of Proposition 2 are verified, and $\tilde{S}$ is a Nash equilibrium.

Concerning the second part, again Proposition 5 implies that any strategy S having $D(S) < D(\tilde{S})$ requires that at least one activity u with $A_u \in \mathcal{A}^-$ must be crashed with respect to $\tilde{S}$. However, since in this case $c_u > w_u\pi$, agent A_u would be willing to get back to $\tilde{S}$, so S would not be a Nash equilibrium. □

5 Conclusion and future research directions

In this paper we have introduced a noncooperative game-theoretic model for a multi-agent project scheduling problem. In the problem considered, a reward is given by the project customer depending on the project makespan. Such reward is shared between the agents according to some predefined ratios. Each agent is responsible for a set of activities, and incurs costs depending on the

activity durations, as in classical PERT/CPM models. In this paper we focused on the concept of Nash equlibrium, and we propose a polynomial-time algorithm to find a Nash equilibrium. We also prove that finding a Nash equilibrium with bounded makespan is in general NP-hard, but it becomes easy if each agent has only one activity. We notice also that the results of this work can be generalized to piecewise linear crashing-cost functions (as proposed in [Bachelet and Mahey, 2003]).

We next sketch a few possible directions for future research.

- In general, algorithm FINDNASH (Section 4.2) may require the solution of $m(m+1)/2$ linear programs. An open question is whether a more efficient algorithm can be devised to compute a Nash equilibrium.

- While in this paper we mainly concentrated on the concept of Nash equilibrium, another relevant concept in a multi-objective setting is that of *Pareto optimal strategies*, i.e., those for which no other strategy exists, which is strictly better for at least one agent and not worse for all other agents. Future research may address, among other issues, the characterization of Pareto optimal strategies and its relationship with Nash equilibria. It is possible to show that in general a strategy can be both Pareto optimal and a Nash equilibrium, just one of the two, or neither Pareto optimal neither Nash equilibrium.

- Another research direction is the design of distributed cooperation mechanisms, aimed at bringing the agents to collectively reach a stable strategy. Indeed, the study of such mechanisms may be important in real cooperation contexts, in which a central agent having complete information on the system does not exist. In such a context, several important issues should be considered, including partial or uncertain agents' knowledge, limited cooperation among the agents, procedural issues (such as the sequence in which agents make their decisions) etc.

- Enlarging the scope of the model, one might also analyze the scenario in which the values w_u are not predetermined during the design phase, but are rather another output of the optimization phase.

Acknowledgements

This work was supported by the ANR project no. 08-BLAN-0331-02 named "ROBOCOOP". The authors wish to acknowledge two anonymous reviewers for their remarks that significantly helped improving the paper.

References

[Adhau et al., 2012] Adhau, S., Mittal, M. L., and Mittal, A. (2012). A multi-agent system for distributed multi-project scheduling: An auction-based negotiation approach. *Engineering Applications of Artificial Intelligence*, 25(8):1738–1751.

[Agnetis et al., 2004] Agnetis, A., Mirchandani, P. B., Pacciarelli, D., and Pacifici, A. (2004). Scheduling problems with two competing agents. *Operations Research*, 52:229–242.

[Ahuja et al., 1993] Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993). *Network Flows*. Prentice-Hall, Upper Saddle River, NJ, USA.

[Bachelet and Mahey, 2003] Bachelet, B. and Mahey, P. (2003). Minimum Convex-Cost Tension Problems on Series-Parallel Graphs. *RAIRO Operations Research*, 37-4:221–234.

[Brânzei et al., 2002] Brânzei, R., Ferrari, G., Fragnelli, V., and Tijs, S. (2002). Two approaches to the problem of sharing delay costs in joint projects. *Annals of Operations Research*, 109:359–374.

[Ciurea and Ciupalâ, 2004] Ciurea, E. and Ciupalâ, L. (2004). Sequential and parallel algorithms for minimum flows. *Journal of Applied Mathematics and Computing*, 15:53–75.

[Confessore et al., 2007] Confessore, G., Giordani, S., and Rismondo, S. (2007). A market-based multi-agent system model for decentralized multi-project scheduling. *Annals of Operations Research*, 150:115–135.

[Demeulemeester and Herroelen, 2002] Demeulemeester, E. L. and Herroelen, W. S. (2002). *Project Scheduling - A Research Handbook*. Kluwer Academic Publishers, Dordrecht, The Netherlands.

[Estévez-Fernández, 2012] Estévez-Fernández, A. (2012). A game theoretical approach to sharing penalties and rewards in projects. *European Journal of Operational Research*, 216(3):647–657.

[Evaristo and van Fenema, 1999] Evaristo, R. and van Fenema, P. C. (1999). A typology of project management: emergence and evolution of new forms. *International Journal of Project Management*, 17(5):275–281.

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

[Homberger, 2012] Homberger, J. (2012). A (mu,lambda)-coordination mechanism for agent-based multi-project scheduling. *OR Spectrum*, 34(1):107–132.

[Knotts and Dror, 2003] Knotts, G. and Dror, M. (2003). Agent-based project scheduling: Computational study of large problems. *IIE Transactions*, 35(2):143–159.

[Knotts et al., 2000] Knotts, G., Dror, M., and Hartman, B. C. (2000). Agent-based project scheduling. *IIE Transactions*, 32(5):387–401.

[Lau et al., 2005a] Lau, J. S. K., Huang, G. Q., Mak, K. L., and Liang, L. (2005a). Distributed project scheduling with information sharing in supply chains: part I an agent-based negotiation model. *International Journal of Production Research*, 43(22):4813–4838.

[Lau et al., 2005b] Lau, J. S. K., Huang, G. Q., Mak, K. L., and Liang, L. (2005b). Distributed project scheduling with information sharing in supply theoretical analysis and computational study. *International Journal of Production Research*, 43(23):4899–4927.

[Leung et al., 2010] Leung, J. Y.-T., Pinedo, M. L., and Wan, G. (2010). Competitive two-agent scheduling and its applications. *Operations Research*, 58:458–469.

[Li and Zhou, 2012] Li, Z. and Zhou, H. (2012) Coordination Mechanism of Economic Lot and Delivery Scheduling Problem. *Industrial Engineering Journal*, 15:18–22.

[Phillips and Dessouky, 1977] Phillips, S. and Dessouky, M. I. (1977). Solving the project time/cost tradeoff problem using the minimal cut concept. *Management Science*, 24(4):393–400.

[Wang et al., 2011] Wang, L., Zhan, D., Nie, L., and Xu, X. (2011). Research framework for decentralized multi-project scheduling problem. *2011 International Conference on Information Science and Technology, ICIST 2011*, pages 802–806.