Finding an optimal Nash equilibrium to the multi-agent project scheduling problem

Cyril Briand, Sandra Ulrich Ngueveu, Přemysl Šůcha

Finding an optimal Nash equilibrium to the multi-agent project scheduling problem

Cyril Briand [(1)(2)], **Sandra Ulrich Ngueveu**[(1)(2)] **and Přemysl Šůcha**[(1)(3)]

[(1)] CNRS ; LAAS ; 7 avenue du colonel Roche,

F-31077 Toulouse Cedex 4, France,

[(2)] Université de Toulouse ; UPS, INP; LAAS;

F-31077 Toulouse Cedex 4, France

[(3)] Czech Technical University in Prague; Faculty of Electrical Engineering;

Karlovo namesti 13, 121 35, Prague 2, Czech Republic

Emails: {briand, ngueveu}@laas.fr suchap@fel.cvut.cz

Cooperative projects involve a set of self-interested contractors, each in charge of a part of the project. Each contractor may control the duration of his activities, which can be shorten up to an incompressible limit, by gathering extra resources at a given cost. In this context the resulting project makespan depends on all contractors' decisions. The client of the project is interested in a short project makespan. As an incentive, the client offers a daily reward to be shared among the contractors in order to complete the project earlier than expected. In practice, either the reward sharing policy results from an upfront agreement or payments are freely allocated by the client himself. Each contractor is only interested in the maximization of his own profit, and behaves accordingly. This paper addresses the problem of finding a Nash equilibrium and a sharing policy that minimize such project makespan while ensuring its local stability. We explain how the resulting problem, which is NP-hard, can be modeled and solved with mixed integer linear programming (MILP). A computational analysis on large instances proves the effectiveness of our approach. Useful insights are also derived from an empirical investigation of the influence of reward sharing policy, for a better understanding of how a project customer should make the most of his funds in such project management context.

Key words: project scheduling; time/cost trade-off; Nash equilibrium; mixed integer programming

1. Introduction

Time-Cost tradeoffs in project management have been widely studied in the literature De et al. (1995), Diaby et al. (2011). The objective is often either to minimize the project makespan at a minimum cost, or to maximize a net present value. Recently, the focus has been put on the dynamics between the client and a contractor in charge of the realization of the project, through mechanisms

such as time-dependent bonuses, penalties and payment retention Bey et al. (1981),Dayanand and Padman (2001),Bahrami and Moslehi (2012). However, most results in this area considered only one-side's benefit, either the client or the contractor. In his study of the definition of payments schedules, Szmerekovsky (2005) highlighted this gap and emphasized on the necessity to consider the reaction of one to the decisions of the other: the client may be interested in a short project makespan, but the contractor is self-interested and would behave accordingly by readjusting the processing and completion time for each activity to maximize his own financial objective. Szmerekovsky and Venkateshan (2012) extended this work to irregular time-cost tradeoffs. Unfortunately, such studies consider interactions with a single contractor. In practice, different contractors would often be involved, each in charge of a part of the project. The project output depends on all contractors' decisions, yet each one behaves in its own best interest. In this case, the issue becomes twofold: how to get the shorter makespan the client aims at, and how to reach an equilibrium among the different contractors to ensure the stability of the resulting schedule.

In this paper we consider a cooperative project which involves a set of self-interested entities called *agents*, each in charge of a part of the project. The project is classically defined by an acyclic activity network. Each agent is able to reduce the duration of his activities by gathering extra resources, at a given cost. Consequently, the project makespan depends on the decisions (called strategies) of all agents. Since the project *customer* is interested in reducing project makespan, he offers a reward to be shared among agents if the project ends earlier than expected. The amount of the reward is given by the number of saved days, i.e. for each saved day the customer offers a constant *daily reward*. Daily reward sharing is part of the decision. Moreover, every agent has to choose his individual strategy by specifying the durations of his activities, which have minimum and maximum values. Activity duration shortening causes expenses on agent's side called *crashing cost* (or compression cost). Each agent wants to maximize his profit equal to the difference between his part of the reward and his expenses for shortening activities. We highlight that, in the single-agent case, this framework exactly corresponds to the well known "Project Time/Cost Trade-off problem"; which received a lot of attention in PERT/CPM methodology Demeulemeester and Herroelen (2002).

A strategy profile is the concatenation of all agents' individual strategies. It is considered locally *stable* if there is no incentive for any agent to modify his own strategy in order to improve his profit. The stability of a strategy is important since it ensures that agents can trust each other. It is totally connected to the notion of a Nash equilibrium. Therefore, from the customer viewpoint, the problem of finding a Nash equilibrium that minimizes the project makespan is of interest, because its solution gives the lower makespan that can be reached for the customer, provided that the organization remains stable. Hence the objective is to determine the duration of activities

and the sharing of the reward in order to find a stable schedule with minimal project makespan. This problem generalizes the cooperative project scheduling with controllable processing times introduced in Briand and Billaut (2011) by considering variable sharing policies.

An example of this problem is depicted in Figure 1 with an activity-on-node network involving five activities $a, \ldots, e$ (activities 1 and 7 are dummy). The customer offers a \$120 reward which is shared equally between agents 1 and 2. The amount of money associated to each node corresponds to agent expenses for decreasing the activity duration by one time unit. For example, to decrease by one unit the duration of activity c, agent 1 has to pay \$20. In the case this decision induces a decrease of the project duration, he will obtain a \$60 reward from the customer and therefore, it is profitable for him to do so (i.e., the original solution was not stable). Furthermore, if agent 2 is able to increase the project duration by one unit by increasing duration of activities d and e, then the original solution is not stable as well. Indeed, by increasing duration of his activities agent 2 saves \$70, while the reward offered by the customer is only \$60. Now, if we offer a bigger portion of the daily reward to agent 2, then the solution may become stable. Fundamentally, the stability of this problem depends on the capability of agents to decrease/increase the duration of selected activities and on the difference between agent cost and offered reward.

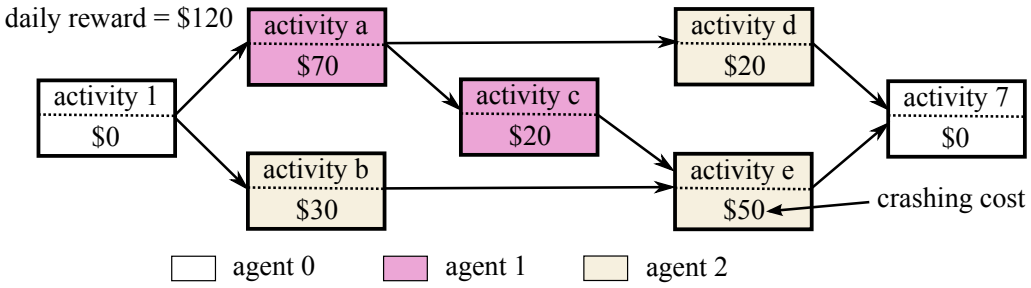


Figure 1 **A project represented by an activity on node network**

Briand and Billaut (2011) described several applications of such problems, from the fields of building trades, supply chain networks and automotive industries Diaz (2006), Dudek and Stadtler (2005), where different actors, for example regrouped in a consortium, must cooperate but have their own goal/agenda. The authors in Briand and Billaut (2011) also touched upon the notions of solution efficiency and stability. They define an efficient solution as a Pareto optimal solution and a stable solution as a Nash equilibrium.

The remainder of the paper is organized as follows. The next section reviews related works and states this paper contribution. In Section 3 we formally define the problem addressed in this paper. Section 4 discusses the complexity analysis of a very similar problem with weaker stability constraints. Section 5 presents a formal mathematical model of the addressed problem which is

transformed into a MILP formulation in Section 6. Properties of reward sharing are discussed in Section 7. Experimental results are discussed in Section 8 and the last section concludes this work.

2. Related Works

To the best of our knowledge, only two papers address the problem outlined in Section 1, considering the special case with fixed sharing policies. This special case was introduced in Briand and Billaut (2011) for the first time. Agnetis et al. (2012) proved that the problem with fixed sharing policies is NP-hard. They proved that a strategy profile is a Nash equilibrium if and only if there does not exist any increasing or decreasing residual cut which allows one agent to increase his profit with respect to the daily reward. This notion of a cut will be further developed in the rest of the paper.

Recently, some papers tackled the issue of Nash equilibrium for different scheduling problems. Averbakh (2010), studies the problem of scheduling activities of a project for a firm that competes with another one that has to perform the same project. The profit that a firm gets from each activity depends on whether the firm finishes the activity before or after its competitor. The objective of that problem is to find a Nash equilibrium solution. Lee et al. (2012) take interest in coordination mechanisms for the distributed scheduling of n jobs on m parallel machines, where each agent selects a machine to process each of his jobs. Without a central authority to construct a schedule, each agent acts selfishly to maximize his own utility. However, the overall system performance is measured by a central objective which is different from the agents' objective. In that paper the authors give bounds for the price of anarchy defined as the maximum ratio of the objective function value of a Nash equilibrium schedule versus that of an optimal coordinated schedule. Estévez-Fernández (2012) deals with a cooperative project game. The paper analyzes situations in which a project consisting of several activities is not realized as initially planned. If the project is expedited, a reward arises. On the other hand, a penalty arises if the project is delayed. This paper is focused on how to divide the total post-project reward (or penalty) among the activities.

Works dealing with resources sharing in multi-agent systems where the objective is to find Pareto optima are more frequent. Recently, Elvikis and T'kindt (2012) studied a two-agent scheduling problem on uniform parallel machines. They consider all jobs have the same processing time but the speeds of machines are different. They assume that agents are selfish and their criteria are conflicting due to resources sharing with the other agent. The authors propose a polynomial time algorithm for the enumeration of the strict Pareto optima. Authors in Agnetis et al. (2004) also deal with a scheduling problem considering two competing agents. They consider a single machine scheduling environment, each agent managing his set of jobs and having his objective function, which depends on the completion times of his jobs only. This work addresses the complexity of various problems related to Pareto-optimal schedules set. Decentralized multi-project scheduling

is considered in Homberger (2012). In this problem many distributed projects, sharing a set of resources, have to be planned simultaneously by a group of local decision makers having their own objective functions. The authors describe a coordination mechanism for managing the usage of shared resources.

This paper considers the problem of finding a Nash equilibrium that minimizes the project makespan, including when the customer can decide how to share the reward among the agents. We further refer to this problem as $\mathcal{P}_1$. The key contributions are the following: the paper (i) provides an optimal mixed integer linear programming (MILP) model for problem $\mathcal{P}_1$ using the simplified conditions, (ii) extends the network flow model proposed by Phillips and Dessouky (1977) for Project Time/Cost Trade-off problem to this multi-agent problem, (iii) shows complexity analysis of two sub-problems where the notion of Nash equilibrium is substituted by a weaker constraint (problem denoted as $\mathcal{P}_2$), (iv) discusses the influence of the reward sharing policy and derives useful insights which should help project customers make the most of their funds in a project management context and (v) provides a computational analysis of the MILP model on large instances to prove the effectiveness of the approach.

3. Problem Statement

3.1. The Multi-agent Project Scheduling Problem

The multi-agent project scheduling environment is defined as a tuple $\langle \mathcal{G}, \mathcal{A}, \underline{P}, \overline{P}, C, \pi, w \rangle$ where $\mathcal{G}$ is the project network, $\mathcal{A}$ represents the set of agents, $\underline{P}, \overline{P}, C$ are parameters of project activities and π, w represent the daily reward offered by the customer. The project network is an activity-on-arc graph $\mathcal{G} = (X, U)$ where U is the set of activities and X is the set of project events. In the set of nodes $X = \{1, \ldots, n\}$, nodes 1 and n correspond to the project beginning and project end. The set of arcs U can be decomposed into two subsets: U_R and U_D, corresponding respectively to the real and *dummy* project activities. Such dummy activities are needed to correctly represent all precedence constraints Demeulemeester and Herroelen (2002). U_R is distributed among the given set $\mathcal{A} = \{A_1, \ldots, A_m\}$ of m agents, and $\mathcal{T}_u$ denotes the set of activities assigned to agent A_u. Each activity $(i,j) \in U_R$ belongs to exactly one agent, i.e., $\mathcal{T}_u \cap \mathcal{T}_v = \emptyset$ for any pair of agents $(A_u, A_v) \in \mathcal{A}^2$ such that $u \neq v$.

Every activity $(i,j) \in U$ has a *minimal duration* $\underline{p}_{i,j} \in \underline{P}$, a *normal duration* $\overline{p}_{i,j} \in \overline{P}$ and a *unitary crashing cost* $c_{i,j} \in C$. For each activity $(i,j) \in U_R$, the processing time $p_{i,j}$ belongs to $\left[\underline{p}_{i,j}, ..., \overline{p}_{i,j}\right] \cap \mathbb{Z}$. It is an integer variable and its value is chosen by agent A_u who owns it (meaning $(i,j) \in \mathcal{T}_u$). The crashing cost incurred by agent A_u when shortening activity $(i,j) \in \mathcal{T}_u$ from $\overline{p}_{i,j}$ to $p_{i,j}$ is $(\overline{p}_{i,j} - p_{i,j})c_{i,j}$. For each dummy activity $(i,j) \in U_D$, we have $\underline{p}_{i,j} = \overline{p}_{i,j} = 0$ and $c_{i,j} = 0$. It is noteworthy that this linear crashing cost function can be easily generalized to piecewise linear crashing cost (e.g., see Bachelet and Mahey (2003)).

The *strategy* of an individual agent is the duration vector P_u of activities $\mathcal{T}_u$. An agents' *strategy profile* S is a vector that gathers all individual agent strategies, i.e., $S = (P_1, \ldots, P_m)$. For each strategy profile S, the project makespan $D(S)$ corresponds to the longest path in the graph $\mathcal{G}(S)$ where length of arc (i,j) equals $p_{i,j}$. Let w_u be the percentage of the daily reward π allocated to agent A_u. The *profit sharing policy* chosen by the customer must verify $\sum_{A_u \in \mathcal{A}} w_u = 1$, with $w_u \in [0,1]$. The resulting profit $Z_u(S)$ of an agent A_u, when strategy profile S is applied, can be computed with equation (1), where $(\overline{D} - D(S))$ is the project makespan reduction and $\overline{D}$ is the maximal project duration given by the longest path in the project network $\mathcal{G}$ when no activity is crashed (i.e., when $p_{i,j} = \overline{p}_{i,j},\ \forall (i,j)$).

$$Z_u(S) = w_u \pi \left(\overline{D} - D(S)\right) - \sum_{(i,j) \in \mathcal{T}_u} c_{i,j}(\overline{p}_{i,j} - p_{i,j}), \tag{1}$$

Assuming that each agent aims at maximizing his profit, the multi-agent project scheduling problem can be formulated as the following multi-objective optimization problem

$$\begin{aligned}
\max_{\forall A_u \in \mathcal{A}} \ & \left\{ w_u \pi \left(\overline{D} - (t_n - t_1)\right) - \sum_{(i,j) \in \mathcal{T}_u} c_{i,j}(\overline{p}_{i,j} - p_{i,j}) \right\} \\
s.t. \ \ & t_i + p_{i,j} \leq t_j && \forall (i,j) \in U, \\
& \sum_{A_u \in \mathcal{A}} w_u = 1, \\
& p_{i,j} \in \left[\underline{p}_{i,j}, \ldots, \overline{p}_{i,j}\right] \cap \mathbb{Z} && \forall (i,j) \in \mathcal{T}_u, \\
& t_i \geq 0 && \forall i \in X, \\
& 0 \leq w_u \leq 1 && \forall (i,j) \in U,
\end{aligned}$$

where, $\forall i \in X$, t_i is the occurrence time of project event i.

The objective in this paper is to determine an optimal S^* strategy profile and a sharing policy w_u^* such that the solution is a Nash equilibrium and the project makespan is minimized. This problem will be further referred to as $\mathcal{P}_1$. Recall that for the special case where w_u are fixed, $\mathcal{P}_1$ is known to be NP-hard Agnetis et al. (2012). On the other hand, considering one agent only, $\mathcal{P}_1$ reduces to the famous Project Time/Cost Trade-off problem Demeulemeester and Herroelen (2002), which is polynomially solvable.

3.2. Problem Properties

In general, a Nash equilibrium is a stable strategy profile such that no agent is interested in changing his strategy. Let S_{-u} denote the strategies played by the $(m-1)$ agents except A_u (i.e.,

$S_{-u} = (P_1, \ldots, P_{u-1}, P_{u+1}, \ldots P_m)$). Then, focusing on a particular agent A_u, S can be also written as a couple (S_{-u}, P_u).

Definition 1. For a given sharing policy w_u, a strategy profile vector $S = (P_u, S_{-u})$ is a Nash equilibrium if for all agents A_u and any other strategy P'_u, we have $Z_u(P_u, S_{-u}) \geq Z_u(P'_u, S_{-u})$.

It is obvious that an algorithm looking for a Nash equilibrium can ignore all solutions where an agent can increase his profit (by changing his strategy) without negatively affecting neither the other agents nor the project makespan D. Let us refer to these strictly dominated solutions as *poor strategy profiles*.

Definition 2. A strategy profile S with project duration $D(S)$ and given w_u is said poor if there exists an alternative strategy $S' = (P'_u, S_{-u})$, only differing from S by the strategy taken by A_u and such that $Z_u(S) < Z_u(S')$, $Z_v(S) \leq Z_v(S')$, for all $A_v \neq A_u$ and $D(S') = D(S)$.

A poor strategy profile $S = (P_u, S_{-u})$ can be easily transformed into a *non-poor strategy profile* $S' = (P'_1, ..., P'_m)$ by adapting P_u for each agent $A_u \in \mathcal{A}$, while keeping durations of activities defined by S_{-u}, denoted $p_{i,j}(S_{-u})$, constant. For fixed w, S_{-u} and $D(S) = D(S')$, a non-poor strategy P'_u can be obtained by solving the linear program:

$$
\begin{aligned}
\min \quad & \sum_{\forall (i,j) \in \mathcal{T}_u} c_{i,j} \left(\overline{p}_{i,j} - p'_{i,j} \right) & \\
s.t. \quad & t'_i + p'_{i,j} \leq t'_j & \forall (i,j) \in U, \\
& p'_{i,j} = p_{i,j}(S_{-u}) & \forall (i,j) \in U \setminus \mathcal{T}_u, \\
& t'_n - t'_1 = D(S) & \forall (i,j) \in U, \\
& p'_{i,j} \in \left[\underline{p}_{i,j}, ..., \overline{p}_{i,j} \right] \cap \mathbb{Z} & \forall (i,j) \in \mathcal{T}_u, \\
& t'_i \geq 0 & \forall i \in X.
\end{aligned}
$$

Finding a Nash equilibrium is not easy in general, but using the notions of increasing and decreasing residual cuts, the problem is made easier to tackle Briand et al. (2012). An increasing (resp. decreasing) residual 1-n-cut $\omega = \{\omega^+, \omega^-\}$ is an arc set defined over $\mathcal{G}(S)$ partitioning the set of nodes X into two sets Y and $(X \setminus Y)$ such that $1 \in Y$, $n \in X \setminus Y$, $\omega^+ = \{(i,j) \in U, i \in Y \wedge j \in (X \setminus Y)\}$ and $\omega^- = \{(i,j) \in U, i \in (X \setminus Y) \wedge j \in Y\}$. Then increasing (resp. decreasing) residual 1-n-cut ω is defined as follows.

Definition 3. A residual 1-n-cut ω is called increasing (resp. decreasing): a) if $(i,j) \in \omega^+$ then $p_{i,j} < \overline{p}_{i,j}$ (resp. $p_{i,j} > \underline{p}_{i,j}$); and b) if $(i,j) \in \omega^-$ then $p_{i,j} > \underline{p}_{i,j}$ ($p_{i,j} < \overline{p}_{i,j}$ resp.).

In other words, an increasing residual 1-n-cut represents a set of activities $\omega = \{\omega^+, \omega^-\}$ such that by increasing $p_{i,j} : (i,j) \in \omega^+$ by 1 and by decreasing $p_{i,j} : (i,j) \in \omega^-$ by 1 the project makespan is prolonged exactly by 1. Symmetrically, decreasing residual 1-n-cut represents a set of activities

$\omega = \{\omega^+, \omega^-\}$ such that decreasing $p_{i,j} : (i,j) \in \omega^+$ by 1 and increasing $p_{i,j} : (i,j) \in \omega^-$ by 1 causes a project makespan shortening of exactly 1. These two operations where $p_{i,j}$ is increased (resp. decreased) by 1 are denoted *application* of cut ω on S, i.e., it is a function $\omega(S)$ mapping S onto S'.

In the multi-agent context, it is meaningful to project the crashing cost of the residual 1-n-cut on each agent A_u. Let $cost_u(\omega)$ be the projected crashing cost for agent A_u using 1-n-cut ω. This cost is the amount that agent A_u saves/pays for increasing/decreasing the project makespan by 1 when modifying the durations of tasks $(i,j) \in \mathcal{T}_u \cap \omega$. This price can be expressed as

$$cost_u(\omega) = \sum_{(i,j)\in\mathcal{T}_u\cap\omega^+} c_{i,j} - \sum_{(i,j)\in\mathcal{T}_u\cap\omega^-} c_{i,j}. \tag{2}$$

An increasing cut having maximal $cost_u(\omega)$ is denoted as *optimal increasing cut* $\omega^{inc(u)}$. Similarly, a decreasing cut having minimal $cost_u(\omega)$ is denoted as *optimal decreasing cut* $\omega^{dec(u)}$. Using results presented in Briand et al. (2012), a Nash equilibrium in project network $\mathcal{G}(S)$ can be expressed in this way:

PROPOSITION 1. *For given sharing policy w_u and a non-poor strategy profile S let $\omega^{inc(u)}$ and $\omega^{inc(u)}$ be the optimal decreasing and increasing cuts respectively. Then S is a Nash equilibrium if and only if, for all agents $A_u \in \mathcal{A}$*

$$cost_u(\omega^{dec(u)}) \geq w_u\pi$$

and

$$cost_u(\omega^{inc(u)}) < w_u\pi.$$

The proof of this proposition is detailed in Briand et al. (2012). In this paper, an increasing/decreasing cut ω is said *profitable* for agent A_u if its use leads to a growth of the profit Z_u. Using this term, Proposition 1 can be formulated such that: for given sharing policy w_u a non-poor strategy profile S is a Nash equilibrium if and only if there is no profitable cut for any agent.

3.3. An Illustrative Example

For illustration, let us consider the activity network of Figure 2(a) with 5 activities distributed among agents A_1 and A_2 such that $\mathcal{T}_1 = \{a, c\}$ and $\mathcal{T}_2 = \{b, d, e\}$. Therefore $\mathcal{G} = (X = \{t_1, ...t_4\}, U = \{U_R = \{a, ..., e\}\} \cup \{U_D = \emptyset\})$. For simplicity, we consider fixed w_u and assume that the customer and agents have agreed to share the daily reward $\pi = 120$ fairly: $w_1 = w_2 = 0.5$.

When the durations of all activities are set to their maximal value, i.e., $S = (p_a,\ p_b, p_c, p_d,\ p_e) = (7, 9, 3, 8, 5)$, the project makespan is 15. With the strategy profile $S' = (7, 9, 2, 7, 5)$, the makespan becomes 14 and profits are $Z_1(S') = Z_2(S') = 40$. The makespan can be further reduced to 13 with the strategy profile $S'' = (6, 9, 3, 7, 4)$, leading to $Z_1(S'') = 50$ and $Z_2(S'') = 50$. S'' is efficient

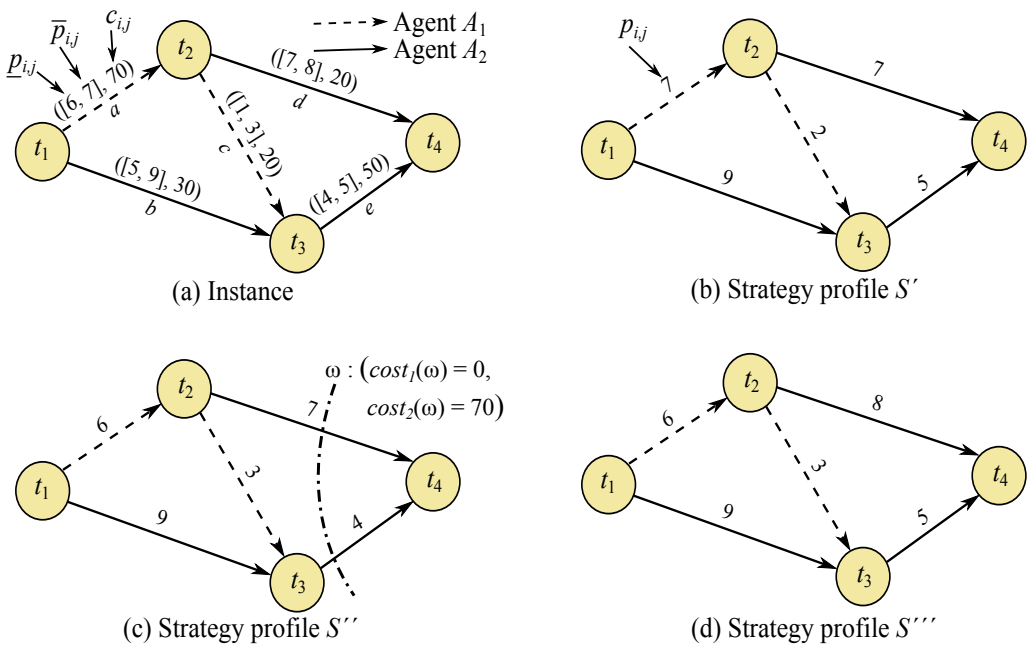


Figure 2 **A multi-agent activity network with two agents and five activities**

because it corresponds to a Pareto optimum that maximizes the profit of both agents. Nevertheless, it is not stable: agent A_2 can improve his profit, to the detriment of A_1, by simply increasing back the duration of d and e, which leads to the strategy $S''' = (6, 9, 3, 8, 5)$, with makespan 14 and profits $Z_2(S''') = 60$ and $Z_1(S''') = -10$. On the other hand, the strategy S', which is not a Pareto optimum, is stable since no agent is able to improve his profit by himself (i.e., it is a Nash equilibrium). Note that it is not possible to find another Nash equilibrium with a shorter makespan, therefore S' is the desired solution.

4. Problem Complexity

Finding a Nash equilibrium that minimizes the project makespan when the profit sharing has been predefined is already NP-hard in the strong sense Briand et al. (2012). Since introducing variable profit sharing generalizes that problem, our problem $\mathcal{P}_1$ is NP-hard as well.

In this section we discuss the complexity of two problems simpler then $\mathcal{P}_1$. We intend to find out which constraints make the problem NP-hard. In both problems we substitute the Nash equilibrium constraint by a weeker constraint stating that the profit of all agents has to be non-negative, i.e., $Z_u(S) \geq 0,\ \forall A_u \in \mathcal{A}$. The first one is the problem of finding minimal makespan when the profit of all agents is non-negative and the duration of activities is a real number. This problem is denoted $\mathcal{P}_2^{\mathbb{R}}$. The second problem considers integer activity durations and is denoted by $\mathcal{P}_2^{\mathbb{I}}$.

Proposition 2. *The multi-agent project scheduling problem which aims at minimizing $D(S)$ under the constraint that agents have nonnegative profits $Z_u(S) \geq 0$, with $p_{i,j} \in \mathbb{R}^+$ (problem $\mathcal{P}_2^{\mathbb{R}}$), can be solved in polynomial time.*

Proof Problem $\mathcal{P}_2^{\mathbb{R}}$ can be described with the following non-linear mathematical model.

$$\min t_n - t_1 \tag{3}$$

s.t.

$$t_j - t_i - p_{i,j} \geq 0 \qquad \forall (i,j) \in U \tag{4}$$

$$\sum_{A_u \in \mathcal{A}} w_u = 1 \tag{5}$$

$$w_u \pi \left(\overline{D} - (t_n - t_1)\right) - \sum_{(i,j)\in\mathcal{T}_u} c_{i,j}(\overline{p}_{i,j} - p_{i,j}) \geq 0 \qquad \forall A_u \in \mathcal{A} \tag{6}$$

where

$$t_i \in \mathbb{R},\ p_{i,j} \in \left[\underline{p}_{i,j}, \overline{p}_{i,j}\right],\ w_u \in [0,1]$$

The model is not linear due to the multiplication $w_u(\overline{D} - (t_n - t_1))$ in constraints (6). The multiplication can be linearized using substitutions $W_u = w_u(\overline{D} - (t_n - t_1))$ where W_u is a new decision variable. Constraints (6) then become $W_u\pi - \sum_{(i,j)\in\mathcal{T}_u} c_{i,j}(\overline{p}_{i,j} - p_{i,j}) \geq 0,\ \forall A_u \in \mathcal{A}$. Consequently, constraint (5) transforms into $\sum_{A_u\in\mathcal{A}} W_u = (\overline{D} - (t_n - t_1))$. The resulting mathematical model has only continuous decision variables and linear constraints, therefore problem $\mathcal{P}_2^{\mathbb{R}}$ can be solved using linear programming in polynomial time. □

Proposition 3. *The multi-agent project scheduling problem ($\mathcal{P}_2^{\mathbb{I}}$) where agents have nonnegative profit $Z_u(S) \geq 0$ and durations are integer $p_{i,j} \in \mathbb{Z}^+$ and $D(S) \leq \lambda$ is strongly NP-complete.*

Proof The complexity of problem $\mathcal{P}_2^{\mathbb{I}}$ can be shown using a reduction from the 3-PARTITION problem which is known to be NP-complete in the strong sense Garey and Johnson (1990). A 3-PARTITION problem consists in deciding whether a given set $\zeta = \{a_1, ..., a_l, ..., a_K\}$ of $K = 3k$ positive integers, such that $\sum_{l=1}^{K} a_l = kB$ and $B/4 \leq a_l < B/2$, can be partitioned into k subsets $\zeta_1, ..., \zeta_u, ..., \zeta_k$ that verify: $|\zeta_i| = 3$ and $\sum_{j\in\zeta_i} a_j = B$, for all $i \in 1, ...k$.

An instance of 3-PARTITION polynomially reduces to an instance of multi-agent project scheduling problem by constructing a project network $\mathcal{G}$ as follows (see Figure 3). Network $\mathcal{G}$ considers $k+1$ agents. Each number $a_l \in \zeta$ is represented by a chain of k activities, with $p_{i,j} \in \left\{\underline{p}_{i,j} = 0, \overline{p}_{i,j} = 1\right\}$ and $c_{i,j} = a_l$, where the order of activities determines their affiliation to the agents, i.e., the first activity belongs to A_1, the second one to A_2 etc. Each first activity of the chain has a predecessor dummy activity starting at node 1. Each last activity of the chain has a successor dummy activity ending at node n. In this way, we have a network composed of K chains connected with the nodes representing the project beginning (node 1) and the project end (node n). Moreover, there is one extra activity from node 1 to node n belonging to dummy agent A_0 with $\underline{p}_{i,j} = \overline{p}_{i,j} = k - 1$ and $c_{i,j} = 0$. This arc represents the project makespan lower bound. The daily reward is $\pi = kB$ and the profit sharing policy is fixed to $w_u = 1/k\ \forall A_u \in \mathcal{A}$.

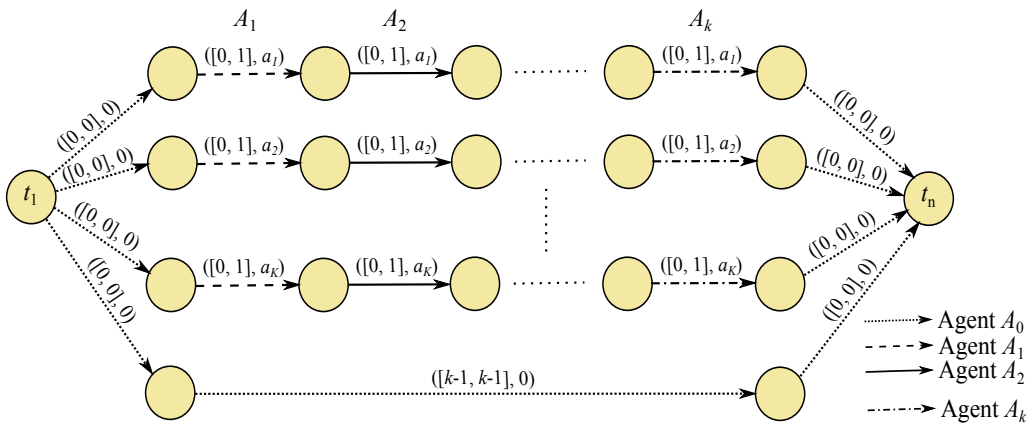


Figure 3 **Reduction from 3-PARTITION problem**

There are only two feasible project makespans $D \in \{k-1, k\}$ on the resulting network, because each chain of activities contains k activities with $\overline{p}_{i,j} = 1$ whereas the extra activity has $\underline{p}_{i,j} = \overline{p}_{i,j} = k-1$. We want to know whether it exists a profitable strategy profile such that $D(S) \leq \lambda$.

If $D(S) = k-1$, it is obvious that exactly one activity of each chain has $p_{i,j} = 0$ while the rest of activities has $p_{i,j} = 1$. If there are two activities on the same chain with $p_{i,j} = 0$ then obviously the solution is poor. In the case $D(S) = k-1$, each constraint (6) transforms into $B \geq \sum_{(i,j)\in\mathcal{T}_u} c_{i,j}(\overline{p}_{i,j} - p_{i,j})$, i.e., the sum of crashing costs of activities with $p_{i,j} = 0$ belonging to agent A_k is smaller or equal than B. Since it holds for each agent and $\sum_{l=1}^{K} a_l = kB$, then $\sum_{(i,j)\in\mathcal{T}_u} c_{i,j}(\overline{p}_{i,j} - p_{i,j}) = B$. Moreover, since $B/4 \leq a_l < B/2$ there are exactly three activities with $p_{i,j} = 0$ belonging to agent A_u. Therefore, the corresponding strategy profile S is a solution of the 3-PARTITION problem, such that when an activity (i,j) that belongs to agent A_u and to chain l has a duration $p_{i,j} = 0$, then the number a_l belongs to subset ζ_u.

On the other hand, if $D(S) = k$, it means that a strategy profile S' with $D(S') = k-1$ does not exist. Therefore, there is no strategy profile S' such that $\sum_{(i,j)\in\mathcal{T}_u} c_{i,j}(\overline{p}_{i,j} - p_{i,j}) = B$ for each agent A_u and the 3-PARTITION problem has no a feasible solution. □

5. Formal Mathematical Model

In this section, a mathematical formulation is given for solving problem $\mathcal{P}_1$. Recall that $\mathcal{P}_1$ aims at finding a Nash equilibrium that minimizes $D(S)$. The next section shows how it transforms into the MILP. Let us consider the mathematical model (7)-(10). We are going to prove that it solves $\mathcal{P}_1$.

$$\min \left((t_n - t_1) + \frac{\sum_{\forall(i,j)\in U} c_{i,j} \left(\overline{p}_{i,j} - p_{i,j}\right)}{1 + \sum_{\forall(i,j)\in U} c_{i,j} \left(\overline{p}_{i,j} - \underline{p}_{i,j}\right)} \right) \tag{7}$$

s.t.

$$t_j - t_i - p_{i,j} \geq 0 \qquad \forall (i,j) \in U \tag{8}$$

$$\sum_{A_u \in \mathcal{A}} w_u = 1 \tag{9}$$

$$\text{cost}_u(\omega^{inc(u)}) < w_u \pi \qquad \forall A_u \in \mathcal{A} \tag{10}$$

where

$$t_i \in \mathbb{R} \qquad \forall i \in X$$

$$p_{i,j} \in \left[\underline{p}_{i,j}, ..., \overline{p}_{i,j}\right] \cap \mathbb{Z} \qquad \forall (i,j) \in U$$

$$w_u \in [0,1] \qquad \forall A_u \in \mathcal{A}.$$

As before, decision variables t_i are associated with occurrence times of the project events. Variables $p_{i,j}$ and w_u correspond to the strategy profile S and the profit sharing, respectively. Constraints (8) model precedence relations between project activities. Constraint (9) enforces the consistency of the daily reward sharing among the agents. The non-explicit (and thus not linear) constraints (10) impose that no increasing residual 1-n-cut ω with a projected profit $\text{cost}_u(\omega)$ greater or equal to $w_u\pi$ (see Proposition 1) exists in solution S. Decreasing cuts do not need to be bounded since the project makespan is minimized, as explained in Proposition 5.

The objective-function (7) primarily aims at minimizing the project makespan $D(S) = t_n - t_1$. Expression

$$\frac{\sum_{\forall (i,j) \in U} c_{i,j} \left(\overline{p}_{i,j} - p_{i,j}\right)}{1 + \sum_{\forall (i,j) \in U} c_{i,j} \left(\overline{p}_{i,j} - \underline{p}_{i,j}\right)}, \tag{11}$$

being strictly smaller than 1, forces the optimal solution to be non-poor. In fact, it behaves as a secondary objective function ensuring that, for the optimal project makespan D, the algorithm chooses a strategy profile S where $\sum_{\forall (i,j) \in U} c_{i,j} \left(\overline{p}_{i,j} - p_{i,j}\right)$ is minimal. Since processing times of activities are integer numbers, then the optimal project makespan $D(S^*)$ is integer as well. Normalizing $\sum_{\forall (i,j) \in U} c_{i,j} \left(\overline{p}_{i,j} - p_{i,j}\right)$ to be strictly smaller than 1, the second objective can become a penalization of the objective function in the mathematical model. More formally:

Proposition 4. *If S is a solution of mathematical model (7)-(10) then S is a non poor strategy profile.*

Proof Let S be a strategy profile obtained by solving mathematical model (7)-(10). If S is poor, then, by definition, there exists $S' = (P'_u, S_{-u})$, only differing from S by the strategy taken by A_u, such that $Z_u(S) < Z_u(S')$, $Z_v(S) \leq Z_v(S')$, for all $A_v \neq A_u$ and $D(S') = D(S)$. Therefore $\sum_{A_u \in \mathcal{A}} Z_u(S) < \sum_{A_u \in \mathcal{A}} Z_u(S')$. Using equation (1) the total profit can be expressed as

$$\sum_{A_u \in \mathcal{A}} \left(w_u \pi \left(\overline{D} - D(S) \right) - \sum_{(i,j) \in \mathcal{T}_u} c_{i,j} (\overline{p}_{i,j} - p_{i,j}(S)) \right) < \sum_{A_u \in \mathcal{A}} \left(w_u \pi \left(\overline{D} - D(S') \right) - \sum_{(i,j) \in \mathcal{T}_u} c_{i,j} (\overline{p}_{i,j} - p_{i,j}(S')) \right). \quad (12)$$

Since $D(S) = D(S')$, Equation (12) can be further simplified

$$\sum_{(i,j) \in U} c_{i,j} (\overline{p}_{i,j} - p_{i,j}(S)) > \sum_{(i,j) \in U} c_{i,j} (\overline{p}_{i,j} - p_{i,j}(S')). \quad (13)$$

Equation (13) shows that objective function (11) is greater for S than for S'. But it leads to a contradiction with the assumption that the objective function is minimal for S. Therefore S has to be a non poor strategy profile. □

Now, let us show that the mathematical model (7)-(10) solves the problem $\mathcal{P}_1$. Using Proposition 1, we show that minimizing objective function (7), while imposing that there is no profitable increasing residual cut, satisfies that there cannot be a profitable decreasing cut, i.e., the solution of the mathematical model is a Nash equilibrium.

PROPOSITION 5. *If S is a solution of mathematical model (7)-(10) then S is a Nash equilibrium.*

Proof Let S be a non-poor strategy profile obtained by solving mathematical model (7)-(10). Assume that S has an optimal decreasing residual 1-n-cut $\omega_1^{dec(u)}$ such that $cost_u(\omega_1^{dec(u)}) < w_u \pi$. Denote a strategy profile $S' = \omega_1^{dec(u)}(S)$ a solution obtained by application of $\omega_1^{dec(u)}$. Since $D(S') = D(S) - 1$, solution S' must contain a profitable increasing 1-n-cut $\omega_2^{inc(v)}$. This cut is either (i) profitable for agent A_u, i.e. $A_v = A_u$, or (ii) profitable for agent A_v such that $A_v \neq A_u$. In both cases, using $\omega_2^{inc(v)}$, another strategy profile $S'' = \omega_2^{inc(v)}(S')$ can be obtained.

(i) When $\omega_2^{inc(u)}$ is profitable for agent A_u, i.e., $cost_u(\omega_2^{inc(u)}) \geq w_u \pi$, we can express the profit of agent A_u after applying $\omega_2^{inc(v)}$ as follows. $Z_u(S'') \geq Z_u(S') - w_u \pi + cost_u(\omega_2^{inc(u)}) \geq \left(Z_u(S) + w_u \pi - cost_u(\omega_1^{dec(u)}) \right) - w_u \pi + cost_u(\omega_2^{inc(u)}) = Z_u(S) - cost_u(\omega_1^{dec(u)}) + cost_u(\omega_2^{inc(u)})$. Since both 1-$n$-cuts were profitable therefore $cost_u(\omega_1^{dec(v)}) < w_u \pi$ and $cost_u(\omega_2^{inc(u)}) \geq w_u \pi$. Consequently $Z_u(S'') > Z_u(S)$. Since the strategy and therefore profit of other agents stayed the same the objective function (11) is higher for S'' than for S. But it lead to a contradiction with the assumption that the objective function is minimal for S. Therefore S can not contain a decreasing 1-n-cut which is profitable for agent A_u.

(ii) If $\omega_2^{inc(v)}$ is profitable for agent A_v such that $A_v \neq A_u$ then it means that $\omega_2^{inc(v)}$ has to involve some activities that become critical after applying cut $\omega_1^{dec(u)}$. Cut $\omega_1^{dec(u)}$ affects activities belonging to A_u only. Decreasing the project makespan can create new critical 1-n-paths. Activities

that become newly critical can belong to an arbitrary agent but their $p_{i,j} = \overline{p}_{i,j}$. It means that their duration can only be decreased. But this fact cannot increase $cost_u(\omega_2^{inc(v)})$ in S' w.r.t. S. Since there is no profitable $\omega_2^{inc(v)}$ in S therefore there the same condition holds for strategy profile S'. But this is in contradiction with a fact that S' contains an increasing profitable 1-n-cut.

Since there is no profitable decreasing 1-n-cut in S for any agent, this solution states a Nash equilibrium. □

The last proposition shows that our model solves problem $\mathcal{P}_1$ since the mathematical model minimizes $D(S)$.

6. A MILP for $\mathcal{P}_1$

This section focuses on the linearization of constraints (10). We propose to determine $\omega^{inc(u)}$ using a finite number of primal-dual constraints. The explanation of this approach is decomposed into 3 main steps. Subsection 6.1 shows a residual network used to determine increasing cuts. Subsection 6.2 details the additional primal-dual constraints used in the MILP model to determine maximal residual 1-n-cuts from the residual network. Finally, Subsection 6.3 describes how the residual network is modeled in the MILP.

6.1. Residual Network

An increasing 1-n-cut $\omega = \{\omega^+, \omega^-\}$ for a non-poor strategy profile S can be defined as a set of activities such that by increasing $p_{i,j}$ of activities $(i,j) \in \omega^+$ by one and decreasing $p_{i,j}$ of activities $(i,j) \in \omega^-$ by one, it will result into strategy $S' = \omega(S)$ with $D(S') = D(S) + 1$. Of course, due to the bounds $\overline{p}_{i,j}$ (resp. $\underline{p}_{i,j}$), some activity durations may not allow anymore increase (resp. decrease) and therefore, these activities cannot be in ω^+ (resp. ω^-). To determine increasing 1-n-cuts, we consider a residual network where crashing costs $c_{i,j}$ are substituted by forward and backward residual costs $\underline{c}^u_{i,j}$, $\overline{c}^u_{i,j}$.

The *residual network* $\mathcal{N}_u(S)$ with respect to an agent $A_u \in \mathcal{A}$ is a tuple $\langle \mathcal{G}(S), \underline{c}^u_{i,j}, \overline{c}^u_{i,j} \rangle$ where $\mathcal{G}(S)$ is the project activity network with strategy profile S. Element $\underline{c}^u_{i,j} \in \mathbb{R}^+$ and $\overline{c}^u_{i,j} \in \mathbb{R}^+$ are the residual costs. These costs reflect the capabilities of corresponding activities to increase or decrease the length of a *critical* 1-n-*path*. A critical 1-n-path in our problem is a path from node 1 to node n such that the sum of $p_{i,j}$ of activities (i,j) belonging to the path is equal to the project makespan. Activities on a critical 1-n-path are called *critical activities*.

Residual costs, on residual network $\mathcal{N}_u(S)$, are given by equations (14) and (15). For activities $(i,j) \in \mathcal{T}_u$ the values are the same as in Phillips and Dessouky's work Phillips and Dessouky (1977) while for activities $(i,j) \notin \mathcal{T}_u$ we have to consider different values, as explained below. It is obvious that, the price of 1-n-cut ω is then given by

$$cost_u(\omega) = \sum_{(i,j) \in \mathcal{T}_u \cap \omega^+} \underline{c}_{i,j} - \sum_{(i,j) \in \mathcal{T}_u \cap \omega^-} \overline{c}_{i,j},$$

which exactly corresponds to Equation (2). In fact these equations extend Phillips and Dessouky (1977) for Project Time/Cost Tradeoff problem since they use the same residual network in every step of their cut search algorithm without considering agents. Here we extend this approach to the multi-agent problem $\mathcal{P}_1$.

$$\underline{c}^u_{i,j} = \begin{cases} 0 & \text{if } (i,j) \in \mathcal{T}_u : \Big((i,j) \text{ is not critical} \vee p_{i,j} = \underline{p}_{i,j}\Big), \\ c_{i,j} & \text{if } (i,j) \in \mathcal{T}_u : \Big((i,j) \text{ is critical} \wedge p_{i,j} > \underline{p}_{i,j}\Big), \\ 0 & \text{if } (i,j) \notin \mathcal{T}_u. \end{cases} \tag{14}$$

$$\overline{c}^u_{i,j} = \begin{cases} c_{i,j} & \text{if } (i,j) \in \mathcal{T}_u : \big((i,j) \text{ is critical} \wedge p_{i,j} < \overline{p}_{i,j}\big), \\ \infty & \text{if } (i,j) \in \mathcal{T}_u : \big((i,j) \text{ is not critical} \vee p_{i,j} = \overline{p}_{i,j}\big), \\ 0 & \text{if } (i,j) \notin \mathcal{T}_u : (i,j) \text{ is not critical}, \\ \infty & \text{if } (i,j) \notin \mathcal{T}_u : (i,j) \text{ is critical}. \end{cases} \tag{15}$$

A forward residual cost of 0 means that the arc does not matter in the computation of the cost of forward arcs (see Equation (14)). This happens if the corresponding task (i,j) is not on a critical 1-n-path (and thus increasing its duration by 1 cannot impact the project duration) or if $p_{i,j}$ cannot be further increased (because $p_{i,j} = \overline{p}_{i,j}$) or if (i,j) belongs to another agent. Otherwise a cost $\underline{c}^u_{i,j} = c_{i,j}$ reflects the fact that the agent can make a profit by increasing the activity duration since it belongs to a critical 1-n-path and $p_{i,j} \leq \overline{p}_{i,j}$.

In a similar way, a backward residual cost of 0 means that the arc does not matter in the computation of the cost of backward arcs for agent u, and thus can be removed from the residual network, which happens if the task is not on a critical 1-n-path or does not belong to the agent (see Equation (15)). Cost $\overline{c}^u_{i,j} = c_{i,j}$ reflects the fact that the agent can make a profit by decreasing the activity duration. Finally, setting $\overline{c}^u_{i,j} = \infty$ ensures that any 1-n-cut that uses (i,j) as a backward arc has a cost small enough to be disregarded. Indeed such 1-n-cut would have $cost_u < 0$ and thus the Equation (10) would be trivially verified for this cut. This is the case of tasks that do not belong the agent although they are on a critical 1-n-path, or of tasks that belong to the agent but can no longer decrease their duration (already at their minimal duration), or of tasks that belong to the agent but are not on a critical 1-n-path (since all tasks of ω^+ belong to a critical 1-n-path). All cuts containing these arcs must be disregarded in order to satisfy the assumption that increasing by one the duration of tasks in ω^+ and decreasing by one the duration of tasks in ω^- causes a project makespan increase of precisely one. For implementation purpose ∞ is replaced in MILP by $C = \sum_{(i,j)\in U} c_{i,j}$ as an upper bound of any cut cost.

Having defined the residual network $\mathcal{N}_u(S)$, we need to compute its maximum increasing cut $\omega^{inc(u)}$ to obtain the left-hand-side of equation (10).

6.2. Maximal Cut in the Residual Network

The main idea of our MILP model lies in determination of $\text{cost}_u(\omega^{inc(u)})$ in Equations (10) by computing the maximal capacity of a 1-n-cut in $\mathcal{N}_u(S)$. To compute $\text{cost}_u(\omega^{inc(u)})$, our approach uses the well-known Max-Flow-Min-Cut theorem Ford and Fulkerson (1956), which infers that the maximum value of a 1-n-flow equals the minimum capacity of a 1-n-cut, and conversely, the minimum value of an 1-n-flow equals the maximal capacity of a 1-n-cut (defined by Equation (6.1)).

Assuming residual network $\mathcal{N}_u(S)$, equations (16)-(17) define a 1-n-flow $f^u_{i,j} \in \mathbb{R}^+$ in the network while equations (18)-(20) define 1-n-cut, denoted as ω, partitioning the set of nodes X into two subsets Y and $(X \setminus Y)$.

$$\sum_{\forall(k,i)\in U} f^u_{ki} = \sum_{\forall(i,k)\in U} f^u_{ik} \qquad \forall i \in X \setminus \{1,n\}, \forall A_u \in \mathcal{A} \tag{16}$$

$$\underline{c}^u_{i,j} \leq f^u_{i,j} \leq \overline{c}^u_{i,j} \qquad \forall (i,j) \in U, \forall A_u \in \mathcal{A} \tag{17}$$

The definition of a 1-n-cut ω requires binary decision variables $\gamma^u_i, \alpha^u_{i,j}$, and $\beta^u_{i,j}$. Variable $\gamma^u_i = 1$ iff $i \in Y$ (note that $1 \in Y$), else $\gamma^u_i = 0$ (note that $n \notin Y$). Variables $\alpha^u_{i,j}, \beta^u_{i,j}$ specify the orientation of the arc in the 1-n-cut, meaning $\alpha^u_{i,j} = 1$ (resp. $\beta^u_{i,j} = 1$) iff $(i,j) \in \omega^+$ (resp. $(i,j) \in \omega^-$). This is very important, since from the constraint (10) point of view, different arc orientation means different arc cost.

$$\alpha^u_{i,j} - \beta^u_{i,j} - \gamma^u_i + \gamma^u_j \leq 0 \qquad \forall (i,j) \in U, \forall A_u \in \mathcal{A} \tag{18}$$

$$\gamma^u_1 = 1 \qquad \forall A_u \in \mathcal{A} \tag{19}$$

$$\gamma^u_n = 0 \qquad \forall A_u \in \mathcal{A} \tag{20}$$

It is not possible to integrate the minimization of the total network flow directly as a set of linear constraints, but it is possible to add previous constraints (16)-(20) and then force the network flow to be equal to the capacity of a 1-n-cut. Indeed, because of the Max-Flow-Min-Cut theorem, that flow would be forced to be minimal. Since a flow is always less or equal to a cut capacity, such equality can be achieved simply by adding the inequality (21), whose right-hand side corresponds to the 1-n-cut cost in the residual network $\mathcal{N}_u(S)$.

$$\sum_{\forall(1,k)\in U} f^u_{1,k} \leq \sum_{\forall(i,j)\in U} \alpha^u_{i,j}\underline{c}^u_{i,j} - \sum_{\forall(i,j)\in U} \beta^u_{i,j}\overline{c}^u_{i,j} \qquad \forall A_u \in \mathcal{A} \tag{21}$$

The right-hand side of (21) contains multiplications of two variables, namely $\alpha^u_{i,j}\underline{c}^u_{i,j}$ and $\beta^u_{i,j}\overline{c}^u_{i,j}$. Fortunately, since $\alpha^u_{i,j}$ and $\beta^u_{i,j}$ are binary, the multiplication can be easily linearized (see Appendix).

Finally, $\sum_{\forall(1,k)\in U} f^u_{1,k}$ is equal to $cost_u(\omega^{inc(u)})$ and therefore constraints (10) can be replaced in the MILP by

$$\sum_{\forall(1,k)\in U} f^u_{1,k} < \pi w_u \qquad \forall A_u \in \mathcal{A}. \tag{22}$$

6.3. Activities Forming Increasing Cuts

Values of forward and backward residual cost $\underline{c}^u_{i,j}, \overline{c}^u_{i,j}$ in the residual network are defined by equations (14) and (15). In this section, we show how these values can be determined by means of a reformulation of precedence constraints. In the mathematical model, constraints (8) can be substituted by constraints (23) and (24). The objective is to determine a set of activities whose processing times increase or decrease will impact the project makespan. The new formulation requires the introduction of four new decision variables: (i) $x_{i,j} \in \{0,1\}$ is equal to 1 iff processing time $p_{i,j}$ can be increased, (ii) $y_{i,j} \in \{0,1\}$ is equal to 1 iff processing time $p_{i,j}$ can be decreased, (iii) $z_{i,j} \in \{0,1\}$ is equal to 1 iff activity (i,j) is on a critical 1-n-path from node 1 to node n and finally, (iv) $s_{i,j} \in \mathbb{R}^+$ is the *slack time*, i.e., the time difference between the end of the activity (i,j) and event t_j.

$$t_j - t_i - p_{i,j} - s_{i,j} = 0 \qquad \forall (i,j) \in U \tag{23}$$

$$\epsilon - z_{i,j} \leq s_{i,j} \leq \overline{s}_{i,j}\,(1 - z_{i,j}) \qquad \forall (i,j) \in U \tag{24}$$

Equations (23) compute the slack times $s_{i,j}$. Constraints (24) express that $s_{i,j} = 0$ iff $z_{i,j} = 1$, otherwise $s_{i,j} > 0$. In these equations, $\overline{s}_{i,j}$ is an upper bound of $s_{i,j}$, which can be taken equal to the project makespan if no activity is crashed, meaning $\overline{s}_{i,j} = \overline{D}$. On the other hand, constant ϵ corresponds to a minimal slack value (when $z_{i,j} = 0$) that can be obviously set to $1/n$.

A 1-n-path is critical iff every activity (k,l) on the path verifies $z_{k,l} = 1$. This property is obtained by adding constraints (25) and (26) into our model, so as to ensure that anytime $z_{i,j} = 1$, then there exist activities (k,i) and (j,l) such that $z_{k,i} = z_{j,l} = 1$.

$$z_{i,j} \leq \sum_{\forall (k,i) \in U} z_{k,i} \qquad (i,j) \in U : i > 1 \tag{25}$$

$$z_{i,j} \leq \sum_{\forall (j,l) \in U} z_{j,l} \qquad (i,j) \in U : j < n. \tag{26}$$

For determining if activity (i,j) can form an increasing 1-n-cut, we use $z_{i,j}$, but also $x_{i,j}$ and $y_{i,j}$ that are determined by introducing constraints (27) and (28) into our model.

$$x_{i,j} \leq \left(\overline{p}_{i,j} - p_{i,j}\right) \leq \left(\overline{p}_{i,j} - \underline{p}_{i,j}\right) x_{i,j} \qquad \forall (i,j) \in U \tag{27}$$

$$y_{i,j} \leq \left(p_{i,j} - \underline{p}_{i,j}\right) \leq \left(\overline{p}_{i,j} - \underline{p}_{i,j}\right) y_{i,j} \qquad \forall (i,j) \in U. \tag{28}$$

Finally, constraints (29) can be added to eliminate some strictly dominated strategies since if an activity (i,j) is not on a critical 1-n-path, then there is no need to decrease its processing time, i.e., $p_{i,j} = \overline{p}_{i,j}$.

$$z_{i,j} \geq x_{i,j} \qquad \forall (i,j) \in U \tag{29}$$

Given $x_{i,j}$, $y_{i,j}$ and $z_{i,j}$, residual costs $\underline{c}^u_{i,j}, \overline{c}^u_{i,j}$ can be easily computed from equations (14) and (15).

The complete MILP model is summarized in Appendix A. It involves a polynomial number of variables and constraints with respect to n and m. Note that in addition to all variables described in this section, the MILP model also contains auxiliary variables $\phi^u_{i,j}$ and $\varphi^u_{i,j}$ resulting from the linearization of constraints (21). Although the model is quite huge, the performance of the model is good as illustrated in experimental results of Section 8.

7. Reward Sharing Policy

An important decision in the multi-agent project scheduling problem is the sharing policy, corresponding to the vector w_u that specifies how the daily reward from the customer is to be distributed among agents. The MILP model from Section 6 finds the *optimal* sharing policy for a given data set, meaning the best dispatch of the reward, so as to minimize the project makespan whilst ensuring the stability of the solution (Nash equilibrium). Of course, this model can also handle special cases where the sharing policy is fixed: the percentage to be received by each agent is predefined and non-modifiable.

There are at least three main reasons to study fixed policies. First, a better understanding of the resulting problem, which was introduced in Briand and Billaut (2011) and proven NP-hard in Agnetis et al. (2012), can be gained. Second, the agents may already have an agreement or predefined rules, in which case the customer would not be able or allowed to reach directly each individual agent. Third and finally, a customer, in control of the reward sharing or in direct contact with every agent, may want to be advised on clear and common guidelines to be followed by his employees on every project they are in charge of on his behalf. Regardless of the motivation for fixed sharing policies, these can be easily implemented with our MILP model, by assigning the variables w_u to the predefined values $w_u = w_u^{\text{fixed}}$.

In this paper, we consider five fixed sharing policies, to be studied and analyzed: random, egalitarian, activity number-based, total cost-based and available cost-based. The random policy, which consists in choosing w_u values randomly for each agent A_u, serves as a baseline: an important difference between its results and the results from "smarter policies" or even the *optimal* policy, would highlight the need for good sharing policies, whereas a negligible difference in the results would call into question their relevance. The egalitarian policy splits the reward equally between the m agents $\left(w_u = \frac{1}{m}\right)$. The three remaining "smarter" fixed policies take into account the proportional importance of each agent. The activity number-based policy computes $w_u = |\mathcal{T}_u| \,/\, |U_R|$ prioritizing agents in charge of most of the activities. The total cost-based policy computes $w_u = \sum_{\forall (i,j) \in \mathcal{T}_u} c_{i,j} \,/\, \sum_{\forall (i,j) \in U_R} c_{i,j}$ prioritizing agents who pay high crashing costs. The

available cost-based policy computes $w_u = \sum_{\forall(i,j)\in\mathcal{T}_u} c_{i,j}\left(\overline{p}_{i,j} - \underline{p}_{i,j}\right) / \sum_{\forall(i,j)\in U_R} c_{i,j}\left(\overline{p}_{i,j} - \underline{p}_{i,j}\right)$ prioritizing agents that have both high crashing cost tasks and many crashable days available.

Other well known cost or benefit sharing policies from the literature on delayed or expedited project problems Hougaard (1990), Moulin and Shenker (1992), Sprumont (2008) could not be used because in such problems, computations are done a posteriori, when all tasks have already been executed, to share the resulting surplus or extra-cost, whereas in our case we are computing the desired tasks durations a priori. For example, it is not possible to fix our coefficients w_u according to the classical serial surplus sharing mechanism, because this mechanism requires to rank each 1-n-path of the graph in increasing order of remaining slack, which is not possible in our case since we do not know in advance what would be the duration of a path and, even less so if it will be critical or not. It may still be possible to adapt some of the ideas or principles to our problem: for example, we could evaluate the importance of each agent A_u, either by computing the project duration that can be achieved if A_u cratches his activities whilst the other agents do not; or by computing the project duration if all agents except A_u cratch their activities. Then all agents could be ranked in order of importance and some of the classical surplus sharing rules could be applied. However, this approach would require several computations, which defeat the purpose of a simple and practical rule. There is no interest in applying too complex or too elaborate rules that do not even guarantee optimality, because we already have a MILP model that can provide the optimal solution.

The goal when studying the chosen fixed sharing policies is twofold: to get a better understanding of the influence of this parameter on the solution of the problem in order to derive useful insights for project customers, and to deduce simple yet efficient guidelines that customers could use in practice. Nevertheless, an explanation on why fixed sharing policies cannot provide (near-)optimal solutions is provided in Subsection 8.3.5.

8. Experimental Results

The algorithm performance was evaluated on a PC with Windows Server 2008 R2 Enterprise OS, 8 GB of RAM, a processor AMD Phenom II X4 945 of 3.0 GHz CPU. The MILP model was implemented in IBM ILOG CPLEX Optimization Studio 12.3 and solved with the CPLEX solver. The project makespan upper bound was determined using the heuristic from Briand et al. (2012). The objective of this heuristic is not to minimize the project makespan, but to the best of our knowledge, this is the only solution previously available for the addressed problem. Since no standard benchmark instances exist for our problem, we generated new instances (see web page blinded for peer review).

Subsection 8.1 explains how instances were generated, while Subsection 8.2 discusses the efficiency of our MILP model. The remaining subsections focus on the influence of the daily reward and sharing policies to derive useful recommendations for customers.

8.1. Benchmark Instances

Problem instances were built up using RanGen1 Demeulemeester et al. (2003) generator. For each problem size, 100 instances were generated with an Order Strength (OS) value = 0.3. Parameter OS represents the number of precedence relations divided by the theoretical maximum number of precedence relations in the network. As RanGen1 produces activity-on-node networks, we converted them to activity-on-arc networks using the algorithm described in Demeulemeester and Herroelen (2002).

The activity duration generated by RanGen is considered as minimal duration $\underline{p}_{i,j}$. The normal activity duration is then set as $\overline{p}_{i,j} = \underline{p}_{i,j} + rand(20)$, where $rand(20)$ is an integer random number between 0 and 20. The activity crashing cost $c_{i,j}$ is computed as an integer random number between 10 and 200.

Because problem instances are generated from activity-on-nodes networks, the number of activities in the resulting activity-on-arc network varies significantly from an instance to another, even if both have the same number of real activities. Recall that dummy activities are necessary to represent precedence constraints. In the remainder of this section, the term "number of activities" will refer to the number of real activities U_R. Of course, the total number of activities $|U|$ in the network is typically much higher than $|U_R|$.

The (real) activities are assigned to agents randomly. In our instances we consider 5 agents unless it is specified otherwise. The daily reward π is determined with respect to the maximal cut cost $C(\omega_{max})$. This cost is given by equation (2) when affiliations to agents are ignored. It represents the theoretical most expensive cut that can be used to decrease project makespan. The daily reward π is determined such that $\pi / C(\omega_{max}) \leq 1$.

8.2. Analysis of the resolution time of the MILP

To illustrate the influence of the number of real activities U_R on the resolution time, we consider 5 sets of benchmark instances for $|U_R| = \{20, 40, 60, 80, 100\}$ with a fixed daily reward $\pi = 0.05 \cdot C(\omega_{max})$. Each benchmark set consists of 100 problem instances considering 5 agents + 1 virtual agent for dummy activities. The results are summarized in Table 1. It shows the average, the minimal and the maximal CPU time needed to solve the instances for each $|U_R|$. In addition, it also shows the average number of all activities U and nodes X. Two observations can be made: first, the total number of activities is significantly larger than the number of real activities, and second,

$\|U_R\|$ [−]	avg $\|U\|$ [−]	avg $\|X\|$ [−]	avg $CPUtime$ $[s]$	min $CPUtime$ $[s]$	max $CPUtime$ $[s]$
20	50.57	20.28	0.941	0.031	2.901
40	162.40	47.24	2.046	1.248	4.852
60	327.41	76.44	5.908	2.106	11.029
80	538.35	106.15	13.627	7.192	32.402
100	790.85	135.05	54.393	12.309	304.528

Table 1 ILP model resolution time depending on the number of activities

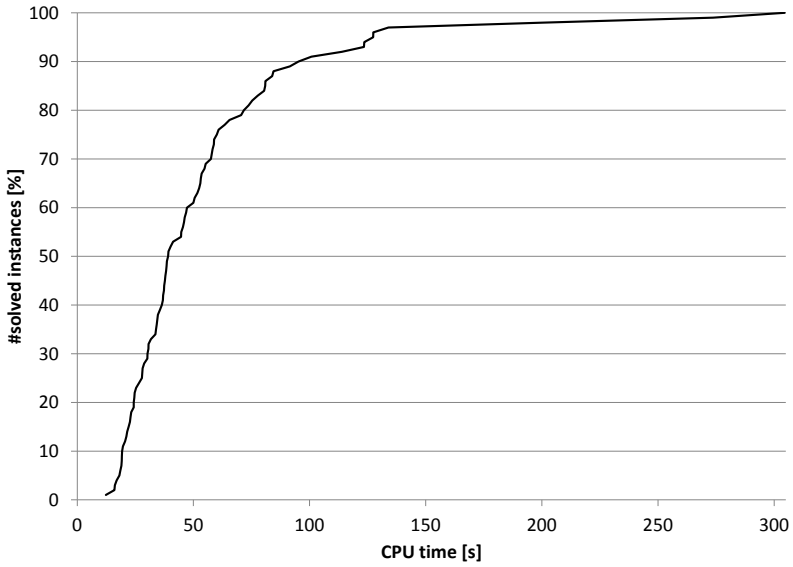


Figure 4 ILP model performance for $n = 100$: percentage of instances solved over time

the solving phase remains very fast since we are able to solve instances with 100 real activities (almost 800 total activities!) in less than 1 minute on average.

Figure 4 presents how many instances (in %) have been solved in a given amount of time for $|U_R| = 100$. The results show that the MILP model is able to solve significantly large instances. Although our MILP model is quite large, this positive behavior can be explained by the fact that the model is composed of several totally unimodular sub-matrices that make the solution search faster. However, the time complexity is quite sensitive to the daily reward parameter, as studied in subsection 8.3.2.

8.3. Analysis of solutions

To compare the sharing policies (optimal, random, egalitarian, activity number-based, total cost-based, and available cost-based) and to analyze the influence of the daily reward, we used the set of benchmark instances with $|U_R| = 60$, 5+1 agents, varied the customer daily reward $\pi = C(\omega_{max}) \times [0,\ 0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.7,\ 1]$, considered the optimal and the five fixed sharing policies from Section 7, then recorded the project makespan, price of stability and CPU time

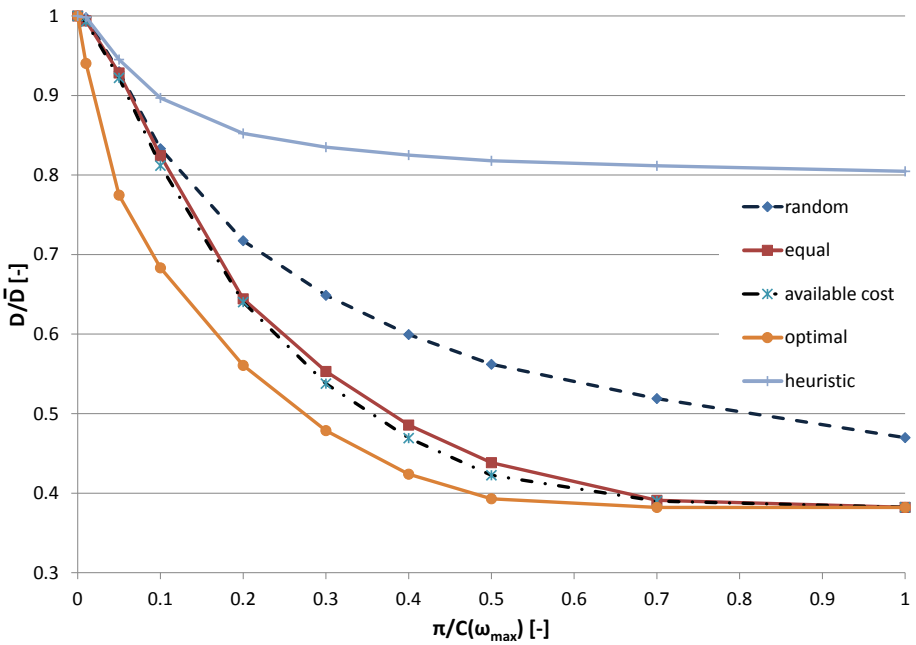


Figure 5 **Project makespan in function of the customer reward**

obtained. Then, all data have been normalized before aggregation to facilitate data mining and ensure the accuracy of subsequent analysis: ratio $D/\overline{D}$ gives the average relative project shortening with respect to the maximal project duration (the lower the better), ratio $\pi/C(\omega_{\max})$ reflects the relative daily reward available (the higher the more money the customer has spent overall).

The results are summarized in Table 2 as well as figures 5 to 7:

- Figure 5 shows the average relative project makespan reduction in function of the relative daily customer reward,
- Figure 6 shows the evolution of the average resolution time in function of the relative daily customer reward,
- Figure 7 shows the evolution of the average price of stability (PoS) in function of the relative daily customer reward. Recall that PoS is defined as a ratio $D(S^*)/D(S^*_{glob})$ where S^* is the optimal Nash solution and S^*_{glob} is the global optimal solution obtained by the MILP model described in the proof of Proposition 2 but considering $p_{i,j} \in \mathbb{Z}$,
- Table 2 details the price of stability, the relative Nash makespan and the relative global optima makespan obtained for the three selected policies, in function of the relative daily reward available.

Results for activity number-based and total cost-based sharing policies are not shown in the graphs since these results are very similar to results of egalitarian sharing policy. The subsequent subsections analyze the results obtained.

8.3.1. Influence of the Daily Reward on the project makespan From the customer point of view, the daily reward π is an important parameter. Of course, it can be expected that the

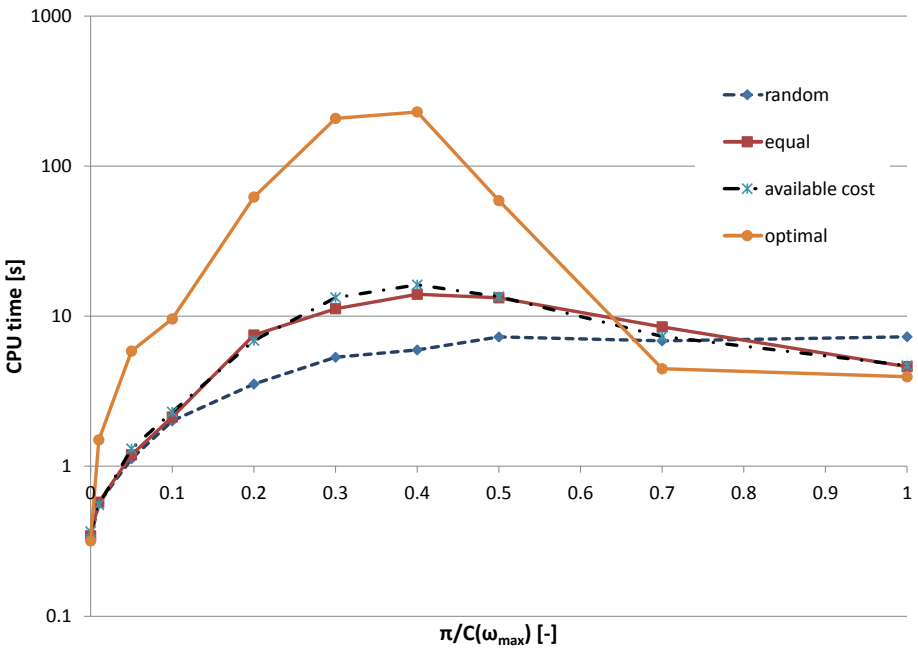


Figure 6 **Resolution time in function of the customer reward**

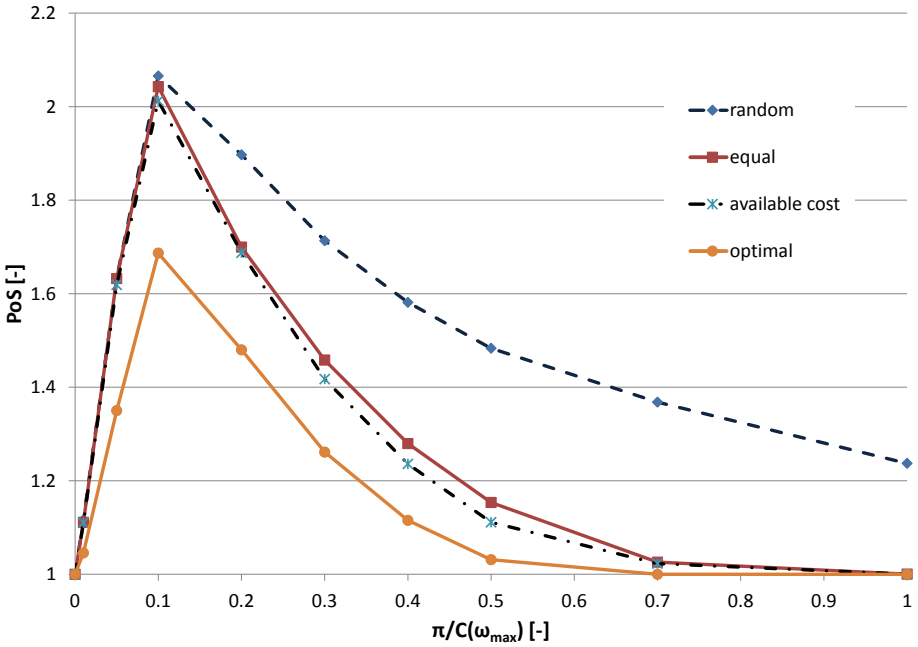


Figure 7 **Price of Stability in function of the customer reward**

larger π, the shorter the project makespan. However, the customer may not want to waste money by paying more than what is necessary for reaching a given makespan, or worse, by paying more without any makespan improvement.

Figure 5 shows the evolution of the project makespan shortening in function of the customer reward for all considered sharing policies. For now, let's focus only on the heuristic and the optimal

	random		available cost		optimal		
$\frac{\pi}{C(\omega_{max})}$ [−]	avg $\frac{D(S^*)}{\overline{D}}$ [−]	avg PoS [−]	avg $\frac{D(S^*)}{\overline{D}}$ [−]	avg PoS [−]	avg $\frac{D(S^*)}{\overline{D}}$ [−]	avg $\frac{D(S^*_{glob})}{\overline{D}}$ [−]	avg PoS [−]
* 0	1.00	1.00	1.00	1.00	1.00	1.00	1.00
0.01	1.00	1.12	0.99	1.11	0.94	0.90	1.05
0.05	0.93	1.63	0.92	1.62	0.77	0.58	1.35
0.1	0.83	2.07	0.81	2.01	0.68	0.41	1.69
0.2	0.72	1.90	0.64	1.69	0.56	0.38	1.48
0.3	0.65	1.71	0.54	1.42	0.48	0.38	1.26
0.4	0.60	1.58	0.47	1.24	0.42	0.38	1.12
0.5	0.56	1.48	0.42	1.11	0.39	0.38	1.03
0.7	0.52	1.37	0.39	1.02	0.38	0.38	1.00
1	0.47	1.24	0.38	1.01	0.38	0.38	1.00

Table 2 Overall sharing policy comparison

policy, the others policies will be analyzed in Section 8.3.4. As expected, the makespan decreases when the reward increases. However, it is important to note that (i) the correlation is not linear, (ii) until approximately $0.3C(\omega_{max})$ every additional reward results into significant makespan decrease, but after $0.3C(\omega_{max})$, each additional reward leads to less and less improvement to the makespan until $0.7C(\omega_{max})$, after which almost no improvement is obtained, of course, (iii) there is a makespan lower bound that cannot be decreased, and finally (iv) the heuristic does not give good results and gives solutions quite far from the optimum.

A project customer should therefore apply our model on the specific instance corresponding to his own project, in order to identify the minimal reward that would lead to the makespan reduction he aims at. Otherwise, a simple yet useful rule of thumb could be not to assign a total reward π higher than 70% of the maximal cut cost.

8.3.2. Influence of the Daily Reward on the resolution time The "optimal" curve of Figure 6 shows the experimental time complexity of our MILP model from the daily reward point of view. It is obvious that for the optimal profit sharing policy the average CPU time is much higher than for fixed policies as the model is more difficult to solve. Also, the model tends to be faster for very high and very low customer rewards, which can be explained by the fact that (i) if the reward is very low, then only a handful of tasks may be considered for crashing because most tasks will have a crashing cost higher than the reward, whereas (ii) if the reward is very high then almost all tasks may benefit from crashing, making the decision problem slightly easier.

Nevertheless, we can conclude that at least up to 60 real activities (size of the instances used), the proposed MILP model is able to solve common problem instances in reasonable time for arbitrary daily rewards.

8.3.3. Influence of the Daily Reward on the price of stability A high PoS means that the makespan could be much lower if the solution was not forced to satisfy the Nash equilibrium condition. It can be expected that PoS would be small for very low and very high customer rewards.

The "optimal" curve of figure 7 shows that PoS is only significant for rewards between $0.1C(\omega_{\max})$ and $0.5C(\omega_{\max})$, and very small otherwise. This interval also corresponds to the customer reward interval in which each reward increase leads to significant makespan reduction on Figure 5.

Overall, the PoS shows which rewards may incite agents to cooperate more, if coalitions and cooperations were considered. Basically, it suggests that for rewards between $0.1C(\omega_{\max})$ and $0.5C(\omega_{\max})$, then agents may have made much more individual profits, had they trusted each other enough to cooperate.

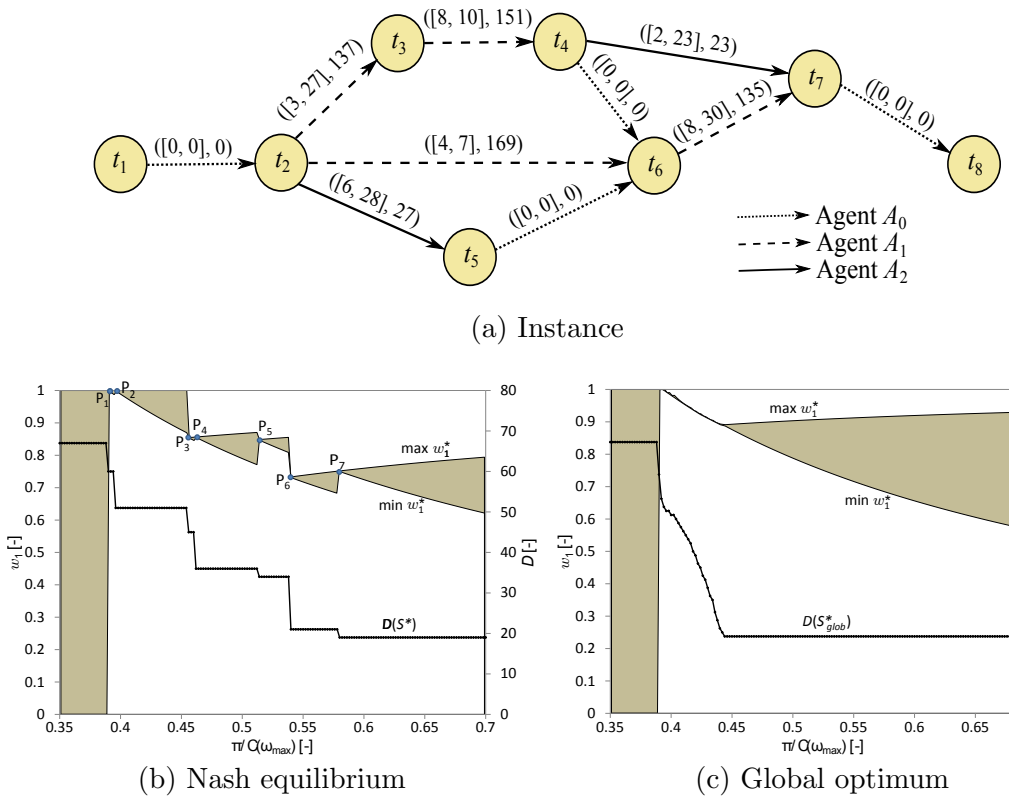
8.3.4. Analysis of the fixed sharing policies In this subsection, the influence of the five fixed sharing policies is analyzed. Figure 5 shows the relative makespan obtained in function of the customer reward. It can be observed that (i) the random policy is the worst of the policies, which highlights the importance and validates the need of "smarter sharing policies", (ii) the random policy produces better solutions than the heuristic from Briand et al. (2012), which confirms that our MILP is a much better resolution tool for the problem than such heuristic, (iii) as expected optimal sharing policy produces better solutions than the fixed sharing policies but the difference is very significant (almost $0.1\bar{D}$), meaning that the optimal sharing provides considerable more savings for the customer, in addition, (iv) surprisingly, the average makespan difference among the "smart" fixed sharing policies is almost negligible, suggesting that there may exist several Nash equilibrium of equivalent makespans, and finally, (v) if one fixed sharing policy had to be chosen, it would be the "available cost" one, which seems to give slightly lower makespan than the others, although the difference is almost negligible.

Figure 6 shows the resolution times depending on the customer reward. Optimal sharing has a significantly higher resolution time compared to fixed policies, which is comprehensible since assigning predefined values to variables w_u makes the problem easier.

Finally, the results in Figure 7 show that optimal sharing policy has considerably lower price of stability than the other policies, which is coherent with the fact that fixed policies lead to higher makespans than optimal sharing.

8.3.5. Focus on (near-)optimal fixed sharing policy Results commented in the previous subsection clearly show that our fixed sharing policies produce results far from the optimum. However, it remains to be proven that these conclusions will hold for every fixed sharing policy a customer could design. This subsection shows a compelling illustrative counter-example which is at the basis of our conjecture that no fixed sharing policy could systematically produce optimal or even near-optimal solutions for different customer rewards and of course different instances.

Lets consider the instance of Figure 8(a) which consists of 6 real activities divided between two agents, plus 4 dummy activities representing precedence constraints. The maximal cut cost



(a) Instance

(b) Nash equilibrium

(c) Global optimum

Figure 8 Optimal profit sharing bounds

$C(\omega_{max})$ can be computed. If a fixed-sharing policy was able to find (near-) optimal solutions, it would accurately evaluate the optimal share w_1^* of agent A_1, for a given customer reward π. The question therefore becomes whether it may be possible to accurately predict w_1^* given the reward, for example by looking for a pattern between the relative total reward $\pi/C(\omega_{max})$ and the optimal share.

Since several solutions may produce the same makespan, therefore for a given reward, we can define an *optimal interval* noted $[\min w_1^*; \max w_1^*]$. The bounds of such interval can be computed by modifying the objective-function of our MILP to respectively $\min(D + w_1/2)$ or $\min(D - w_1/2)$. Note that in this case, the part of the objective-function supposed to ensure that the solution is a non-poor strategy is omitted, since the objective of this experiment is only to compute the optimal makespan $D(S^*)$, and the optimal bounds $\min w_1^*$, $\max w_1^*$, but not the strategy profile.

Figures 8(b) and (c) represent (i) in grey the optimal interval for agent A_1 in function of the relative reward, (ii) with the black line, the corresponding evolution of the project makespan. The difference between these two pictures is that figure (b) shows results for the optimal Nash equilibrium and the figure (c) the global optimal solution. From a customer point of view, it is useless to increase reward if it does not translate into a project makespan decrease, which means that only the 7 highlighted points matter (P1, ..., P7 on the figure). Of course, using the optimal

policy in our MILP model allows us to always find these points. The question remains whether a fixed-sharing policy could find those points as well. It is clear on the Figure 8(b) that on these seven points $\min w_1^* = \max w_2^*$, which means there is no margin for error in the computation of the optimal shares at these points. A fixed sharing policy aspiring at finding near-optimal solutions would have to obtain these five almost exact values. Yet, no simple relation exists between the optimal shares w_1^*, and the customer reward $\pi/C(\omega_{max})$. Sometimes the former decreases when the latter increases. For example, $\pi/C(\omega_{max})$ at P1 is smaller than $\pi/C(\omega_{max})$ at P3, but w_1^* at P1 is higher than w_1^* at P3. Whereas other times the opposite happens. For example, w_1^* at P7 is higher than w_1^* at P6. There is also no simple relation identifiable, on the amount of increase or decrease of w_1^* from a point to another.

These observations lead to the conjecture that no fixed sharing policy can provide (near-)optimal solutions for all customer rewards, even on a single small instance. Moreover, the same experiments performed on the subproblem from Section 4 and illustrated on Figure 8(c) highlight the difference in the level of difficulty when the Nash equilibrium constraint is enforced (figure (b)) compared to when it is disregarded (figure (c)).

The resulting recommendation is to use the proposed MILP to find optimal sharing policies suitable for the specific project of the customer, instead of relying on predefined sharing rules, especially since computational results show that the resolution time is moderate.

9. Conclusion

This paper proposes an optimal MILP-based modeling and resolution method for the multi-agent project scheduling problem where the objective is to find a Nash equilibrium with minimal project makespan. We extend results presented in Briand et al. (2012) and simplify necessary conditions satisfying the solution optimality. The MILP formulation uses a finite number of primal-dual constraints that, up to our knowledge, have not been used in the existing literature yet. Although the MILP model is not trivial and quite large it is able to be solved for instances with more than 100 activities in reasonable time. Another advantage of our MILP formulation lies in the fact that daily reward sharing w_u can be considered as variables without changing the nature of the MILP. It provides a possibility to find even better solutions as is illustrated in the experimental part of the paper. Such a study is of interest in a project management context, especially if the customer is able to influence the rewards sharing policy.

References

Agnetis, A., C. Briand, J. C. Billaut. 2012. The multiagent project scheduling problem with controllable processing times: a game theoretic approach. Tech. rep., LAAS-CNRS.

Agnetis, Allesandro, Pitu B. Mirchandani, Dario Pacciarelli, Andrea Pacifici. 2004. Scheduling problems with two competing agents. *Operations Research* **52**(2) 229–242.

Averbakh, Igor. 2010. Nash equilibria in competitive project scheduling. *European Journal of Operational Research* **205**(3) 552–556.

Bachelet, Bruno, Philippe Mahey. 2003. Minimum convex-cost tension problems on series-parallel graphs. *RAIRO - Operations Research* **37** 221–234.

Bahrami, F., G. Moslehi. 2012. Study of payment scheduling problem to achieve client-contractor agreement. *The International Journal of Advanced Manufacturing Technology* 1–15 Online first, DOI: 10.1007/s00170-012-4023-5.

Bey, R. B., R. H. Doersch, J. H. Patterson. 1981. The net present value criterion: its impact on project scheduling. *Project Management Quarterly* **12** 35–45.

Briand, C., A. Agnetis, J. C. Billaut. 2012. The multiagent project scheduling problem: complexity of finding an optimal nash equilibrium. *13th International Conference on Project Management and Scheduling*. 106–109.

Briand, C., J. Billaut. 2011. Cooperative project scheduling with controllable processing times: A game theory framework. *Emerging Technologies Factory Automation (ETFA), 2011 IEEE 16th Conference on*. 1–7.

Dayanand, N., R. Padman. 2001. Project contracts and payment schedules: the client's problem. *Management Science* **47** 1654–1667.

De, P., E.J. Dunne, J.B. Ghosh, C.E. Wells. 1995. The discrete timecost tradeoff problem revisited. *European Journal of Operations Research* **81** 225–238.

Demeulemeester, E., M. Vanhoucke, W. Herroelen. 2003. Rangen: A random network generator for activity-on-the-node networks. *Journal of Scheduling* **6** 17–38.

Demeulemeester, Erik L., Willy S. Herroelen. 2002. *PROJECT SCHEDULING: A Research Handbook*, chap. 2-1.3.4. Transforming an AoN network into an AoA network with minimal reduction complexity. Kluwer Academic Publishers, 44–48.

Diaby, M., J. M. Cruz, A. L. Nsakanda. 2011. Project crashing in the presence of general non-linear activity time reduction costs. *International Journal of Operational Research* **12** 318–332.

Diaz, L. Martin. 2006. Evaluation of cooperative planning in supply chains: an empirical approach of the european automotive industry. Publications of Darmstadt Technical University 25825, Department of Business Administration, Economics and Law, Institute for Business Studies (BWL).

Dudek, G., H. Stadtler. 2005. Negotiation-based collaborative planning between supply chains partners. *European Journal of Operational Research* **163** 668–687.

Elvikis, Donatas, Vincent T'kindt. 2012. Two-agent scheduling on uniform parallel machines with min-max criteria. *Annals of Operations Research* 1–16 Online first, DOI: 10.1007/s10479-012-1099-0.

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

Ford, L.R., D.R. Fulkerson. 1956. Maximal flow through a network. *Canadian Journal of Mathematics* **8** 399–404.

Garey, M. R., D. S. Johnson. 1990. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co. New York, NY, USA.

Homberger, Jörg. 2012. A (μ, λ)-coordination mechanism for agent-based multi-project scheduling. *OR Spectrum* **34** 107–132.

Hougaard, J. L. 1990. *An Introduction to Allocation Rules*. Springer.

Lee, Kangbok, Joseph Y.-T. Leung, Michael L. Pinedo. 2012. Coordination mechanisms for parallel machine scheduling. *European Journal of Operational Research* **220**(2) 305–313.

Moulin, H., S. Shenker. 1992. Serial cost sharing. *Econometrica* **60** 1009–1037.

Phillips, Steve, Mohamed I. Dessouky. 1977. Solving the project time/cost tradeoff problem using the minimal cut concept. *Management Science* **24** 393–400.

Sprumont, Y. 2008. Nearly serial sharing methods. *International Journal of Game Theory* **37** 155–184.

Szmerekovsky, J. G. 2005. The impact of contractor behavior on the client's payment-scheduling problem. *Management Science* **51** 629–640.

Szmerekovsky, J. G., P. Venkateshan. 2012. An integer programming formulation for the project scheduling problem with irregular time-cost tradeoffs. *Computers and Operations Research* **39** 1402–1410.

Appendix A: ILP Model

$$\min \left((t_n - t_1) + \frac{\sum_{\forall (i,j)\in U} c_{i,j}\left(\overline{p}_{i,j} - p_{i,j}\right)}{1+\sum_{\forall (i,j)\in U} c_{i,j}\left(\overline{p}_{i,j} - \underline{p}_{i,j}\right)} \right) \tag{30}$$

s.t.

$$t_j - t_i - p_{i,j} - s_{i,j} = 0 \qquad \forall (i,j) \in U \tag{31}$$

$$\sum_{A_u \in \mathcal{A}} w_u = 1 \tag{32}$$

$$\epsilon - z_{i,j} \leq s_{i,j} \leq \overline{s}_{i,j}\,(1 - z_{i,j}) \qquad \forall (i,j) \in U \tag{33}$$

$$z_{i,j} \leq \sum_{\forall (k,i)\in U} z_{k,i} \qquad (i,j) \in U : i > 1 \tag{34}$$

$$z_{i,j} \leq \sum_{\forall (j,l)\in U} z_{j,l} \qquad (i,j) \in U : j < n \tag{35}$$

$$x_{i,j} \leq \left(\overline{p}_{i,j} - p_{i,j}\right) \leq \left(\overline{p}_{i,j} - \underline{p}_{i,j}\right) x_{i,j} \qquad \forall (i,j) \in U \tag{36}$$

$$y_{i,j} \leq \left(p_{i,j} - \underline{p}_{i,j}\right) \leq \left(\overline{p}_{i,j} - \underline{p}_{i,j}\right) y_{i,j} \qquad \forall (i,j) \in U \tag{37}$$

$$z_{i,j} \geq x_{i,j} \qquad \forall (i,j) \in U \tag{38}$$

$$\underline{c}^u_{i,j} = c_{i,j} - (1 - x_{i,j})\, c_{i,j} \qquad \forall (i,j) \in U, \forall A_u \in \mathcal{A}, (i,j) \in \mathcal{T}_u \tag{39}$$

$$\overline{c}^u_{i,j} = c_{i,j} + (2 - y_{i,j} - z_{i,j})\, C \qquad \forall (i,j) \in U, \forall A_u \in \mathcal{A}, (i,j) \in \mathcal{T}_u \tag{40}$$

$$\underline{c}^u_{i,j} = 0 \qquad \forall (i,j) \in U, \forall A_u \in \mathcal{A}, (i,j) \notin \mathcal{T}_u \quad (41)$$
$$\overline{c}^u_{i,j} = C z_{i,j} \qquad \forall (i,j) \in U, \forall A_u \in \mathcal{A}, (i,j) \notin \mathcal{T}_u \quad (42)$$

$$\alpha^u_{i,j} - \beta^u_{i,j} - \gamma^u_i + \gamma^u_j \leq 0 \qquad \forall (i,j) \in U, \forall A_u \in \mathcal{A}, (i,j) \notin \mathcal{T}_u \quad (43)$$
$$\gamma^u_1 = 1 \qquad \forall A_u \in \mathcal{A} \quad (44)$$
$$\gamma^u_n = 0 \qquad \forall A_u \in \mathcal{A} \quad (45)$$

$$\sum_{\forall (k,i) \in U} f^u_{k,i} = \sum_{\forall (i,k) \in U} f^u_{i,k} \qquad \forall i \in X \setminus \{1,n\}, \forall A_u \in \mathcal{A} \quad (46)$$
$$\underline{c}^u_{i,j} \leq f^u_{i,j} \leq \overline{c}^u_{i,j} \qquad \forall (i,j) \in U, \forall A_u \in \mathcal{A} \quad (47)$$

$$\phi^u_{i,j} \leq \alpha^u_{i,j} c_{i,j} \qquad \forall (i,j) \in U, \forall A_u \in \mathcal{A} \quad (48)$$
$$\phi^u_{i,j} \leq \underline{c}^u_{i,j} \qquad \forall (i,j) \in U, \forall A_u \in \mathcal{A} \quad (49)$$
$$\phi^u_{i,j} \geq \underline{c}^u_{i,j} - \left(1 - \alpha^u_{i,j}\right) c_{i,j} \qquad \forall (i,j) \in U, \forall A_u \in \mathcal{A} \quad (50)$$
$$\varphi^u_{i,j} \leq \beta^u_{i,j} 3C \qquad \forall (i,j) \in U, \forall A_u \in \mathcal{A} \quad (51)$$
$$\varphi^u_{i,j} \leq \overline{c}^u_{i,j} \qquad \forall (i,j) \in U, \forall A_u \in \mathcal{A} \quad (52)$$
$$\varphi^u_{i,j} \geq \overline{c}^u_{i,j} - \left(1 - \beta^u_{i,j}\right) 3C \qquad \forall (i,j) \in U, \forall A_u \in \mathcal{A} \quad (53)$$

$$\sum_{\forall (1,k) \in U} f^u_{1,k} \leq \sum_{(i,j) \in U} \phi^u_{i,j} - \sum_{\forall (i,j) \in U} \varphi^u_{i,j} \qquad \forall A_u \in \mathcal{A} \quad (54)$$

$$\sum_{\forall (1,k) \in U} f^u_{1,k} < \pi w_u \qquad \forall A_u \in \mathcal{A} \quad (55)$$

where

$t_i, D \in \mathbb{R}$, $s_i \in \mathbb{R}$, $p_{i,j} \in \mathbb{Z}$, $\underline{p}_{i,j} \leq p_{i,j} \leq \overline{p}_{i,j}$,
$0 \leq w_u \leq 1$, $x_{i,j}, y_{i,j}, z_{i,j} \in \{0,1\}$,
$\alpha^u_{i,j}, \beta^u_{i,j}, \gamma^u_i \in \mathbb{Z}^+$, $f^u_{i,j} \in \mathbb{R}^+$,
$\underline{c}^u_{i,j}, \overline{c}^u_{i,j} \in \mathbb{R}^+$, $\phi^u_{i,j}, \varphi^u_{i,j} \in \mathbb{R}^+$, $\epsilon = 1/n$,
$C = \sum_{(i,j) \in U} c_{i,j}$, $\overline{s}_{i,j} = \overline{D}$.