ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Šidlovský** Jméno: **Marko** Osobní číslo: **434962**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Studijní obor: **Softwarové inženýrství**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Nashova rovnováha v rozvrhování projektů s milníky

Název diplomové práce anglicky:

Nash Equilibria in the multi-agent project scheduling problem with milestones

Pokyny pro vypracování:

This thesis addresses a project scheduling problem with subcontractors and milestones where the objective is to find a stable solution with minimal makespan. Stability of the solution is defined via Nash equilibria that guarantees that subcontractors do not have the incentive to deviate from the proposed schedule. The particular objectives of the thesis are:
1) Review the existing works in the project scheduling domain.
2) Devise an exact scheduling algorithm based on lazy constraints generation approach.
3) Implement the scheduling algorithm.
4) Adopt benchmark instances described in [2] and use them for the algorithm benchmarking. Furthermore, investigate the values of the price of anarchy and the price of stability in a large sample of realistic size problems and get useful insights for the project manager.

Seznam doporučené literatury:

[1] Agnetis, A. - Briand, C. - Billaut, J.C. - Šůcha, P. Nash Equilibria for the multi-agent project scheduling problem with controllable processing times In: Journal of Scheduling. 2015, vol. 18, p. 15-27.
[2] Briand, C. - Ngueveu, S.U. - Šůcha, P. Finding an optimal Nash equilibrium to the multi-agent project scheduling problem In: Journal of Scheduling. 2017, 20(5), 475-491.
[3] Estevez-Fernandez, A game theoretical approach to sharing penalties and rewards in projects. In: European Journal of Operational Research. 2012, 216(3), 647--657.

Jméno a pracoviště vedoucí(ho) diplomové práce:

doc. Ing. Přemysl Šůcha, Ph.D., katedra řídicí techniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **05.02.2019** Termín odevzdání diplomové práce: **24.05.2019**

Platnost zadání diplomové práce: **20.09.2020**

doc. Ing. Přemysl Šůcha, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. _____
Datum převzetí zadání

_____
Podpis studenta

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science

Nash Equilibria in the multi-agent project scheduling problem with milestones

Master's thesis

Bc. Marko Šidlovský

Master programme: Open Informatics
Specialization: Software Engineering
Supervisor: doc. Ing. Přemysl Šůcha, Ph.D.

Prague, May 2019

Thesis Supervisor:
doc. Ing. Přemysl Šůcha, Ph.D.
Department of Control Engineering
Faculty of Electrical Engineering
Czech Technical University in Prague
Technická 2
160 00 Prague 6
Czech Republic

Declaration

I hereby declare that I have written this master's thesis independently and quoted all the sources of information used in accordance with methodological instructions on ethical principles for writing an academic thesis. Moreover, I state that this thesis has neither been submitted nor accepted for any other degree.

In Prague, May 2019

.....
Bc. Marko Šidlovský

Abstract

Project scheduling often involves multiple contractors, who are in charge of activities in the project plan. They have the power to decrease the duration of their activities from normal duration to the incompressible limit. The project manager is responsible to deliver the project on time. He specifies the milestones with appropriate due dates and penalties in the project plan. The thesis aims to find a stable solution with minimal project duration. In a stable solution, no contractor has the interest to change the duration of his activities to reduce his expenses, since all other contractors do not change their strategies. We propose a mixed integer linear program formulation with lazy constraint generation for its calculation. Computation analysis confirms the effectiveness of our approach. We investigate the values of the price of anarchy and the price of stability to get useful insight for the project manager.

Keywords: Multi-agent project scheduling, Nash equilibria, Milestones, Mixed integer linear programming

Plánovanie projektov zvyčajne zahŕňa viacerých dodávateľov, ktorí majú na starosti rôzne práce v projektovom pláne. Každý dodávateľ má možnosť skrátiť trvanie svojej aktivity z maximálneho až na minimálny časový limit. Projektový manažér je zodpovedný za včasné dodanie projektu. V projektovom pláne stanovuje míľniky s príslušnými termínmi a pokutami za ich nesplnenie. Cieľom práce je nájsť stabilné riešenie s minimálnym časovým trvaním projektu. V stabilnom riešení nemá žiadny dodávateľ záujem zmeniť trvanie svojich aktivít, aby znížil svoje náklady. To pláti za predpokladu, že všetci ostatní dodávatelia nezmenia svoje stratégie. V práci navrhujeme využitie celočíselného lineárneho programovania s podmienkami generovanými v priebehu programu pre výpočet stabilného riešenia s minimálnym časovým trvaním projektu. Analýza výpočtov potvrdzuje efektívnosť nášho riešenia. Taktiež v práci skúmame ukazovatele v anglickej literatúre označované ako price of anarchy a price of stability, aby sme získali lepšiu predstavu o probléme z pohľadu projektového manažéra.

Kľúčové slová: Rozvrhovanie projektov s viacerými hráčmi, Nashova rovnováha, Míľniky, Celočíselné lineárne programovanie

Acknowledgements

I would like to express my sincere gratitude to my advisor doc. Ing. Přemysl Šůcha, Ph.D. for his patience, motivation, and immense knowledge. His guidance helped me in the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my master's thesis.

List of Tables

List of Figures

List of Acronyms

MAPS Multi-agent project scheduling. 3, 5, 15

MILP Mixed integer linear programming. ix, xi, 4, 13, 15–17, 21–23, 27, 29–31, 34–38, 41

PoA Price of anarchy. ix, 13, 37, 38

PoS Price of stability. ix, 13, 37, 38

Contents

Chapter 1

Introduction

1.1 Motivation

In everyday life, we are planning activities to reach some goals. Our goals can be different, from smaller ones, as for example to organize a weekend family session, to bigger ones like to construct a car or goals to build a skyscraper. In all these situations we have to set up a project plan. The project plan is a set of activities interconnected together, designed to achieve the goal. Each activity has its normal duration, which represents the time for completing it. The project duration, e.g. the project makespan, is the time required to complete all the activities. In some of the projects, we are not the only person responsible to complete the activity. We involve agents (firms, subcontractors, actors, organizations, etc.) to take control of completing particular activities in the project. They also have a chance to decrease the normal duration of the tasks. If they do that, they have to pay crashing cost of the activity. As a project owner, we set up the milestones in the project. Then due date and penalty cost for each milestone force the agents to get the milestones done on time. Managing the project with many activities, agents and milestones can be tricky because all of the agents will try to minimize their expenses. Also changing the strategy of one agent can affect the others.

The objective of our multi-agent project scheduling problem with milestones is to find a stable solution with the minimal makespan. Minimum project makespan is the minimum time required to deliver the project. Stability of the solution is defined via Nash equilibria that guarantee that agents do not have the incentive to deviate from the proposed schedule. Our model is a non-cooperative game where each agent is only interested in minimalization of his expenses - crashing costs of reducing the activities and penalties for tardy milestones.

1.2 Project scheduling assumptions

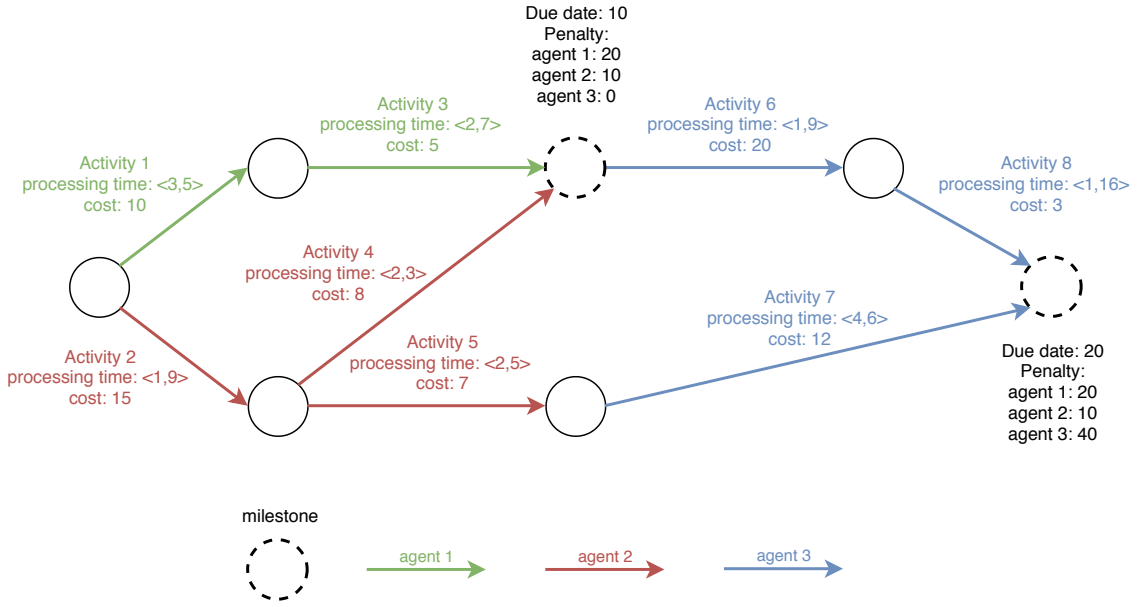


Figure 1.1: A project represented by an activity-on-arch network

Let us expose the problem with more details. You can see an example of a project plan represented by the activity-on-arch network in Figure 1.1. All activities in the project are associated with agents, responsible for concrete activities. An agent sets up a strategy, i.e. time duration of his activities. Final makespan is a union of all agents decisions. The agent can finish his activity in normal duration, or reduce the processing time of the activity to an incompressible limit. Cost of reduction by one-time unit value is crashing cost. The agent is motivated to shorten his activities to reach the milestones on time. If he does not, for each day after the due date he has to pay a penalty. All agents are finding the best strategy for them, where they minimize the sum of crashing costs paid because of shortening the activities and sum of the penalty paid for tardy milestones. Among all solutions where agents minimize their expense (stable solutions), we are looking for the one which minimizes project makespan.

Let us look at Figure 1.1. We can see 8 activities splitted among 3 agents. Agent 1 is responsible for activities 1 and 3, agent 2 for activities 2,4,5 and agent 3 for 6,7,8. Precedence of activities is defined by the graph, e.g. activity 3 can start only after activity 1 is finished. Each activity has normal duration, incompressible limit and crashing cost specified on edges. If agent 1 chooses to reduce processing time of the activity 3 from normal duration 7 to 6, he will pay crashing cost 5. Project owner established 2 milestones in the example. Milestone 1 has due date 10 and penalty 20 for agent 1, 10 for agent 2 and 0 for agent 3. Milestone 2 has a due date 20 and penalty 20 for agent 1, 10 for agent 2 and 40 for agent 3. Due date for milestone 1 is 10, so agent 1 will try to reduce the normal duration of activities 1 and 3 to reach the milestone on time because penalty (20) for the delay is higher than crashing costs

of activity 1 (10) or activity 3 (5). He will try to reduce the processing time of activity 3 by 2-time units and choose to pay 10 instead of 40 for 2 days penalty. For the same reason, agent 2 will reduce the processing time of activity 4, but only by 1-time unit because he gets to an incompressible limit of the activity. This situation also affects agent 1, as he will pay a one-day penalty for milestone 1, so he will choose to reduce the processing time of activity 3 by only a 1-time unit instead of 2. As you can see, the strategy of one agent affects the other agents. Stability of strategy profile depends on the ability of agents to decrease/increase processing times of activities on one hand, and the due dates and penalties of milestones on the other.

1.3 Related Work

We can find different models of the Multi-agent project scheduling (MAPS) problem, where project activities are assigned to a set of agents. Nash equilibria in the MAPS with two agents is described in [1]. When considering the time performance of agents, they compete for a profit.

MAPS can also be studied from resource-constrained project scheduling context point of view, where common limited resources are allocated to agents over time. Finding the best Nash equilibria in a multi-agent resource-constrained project scheduling is presented in [2]. In [3], the authors target mechanisms for allocating shared resources to the agents. In [4], a resource negotiation mechanism is presented and in [5] an auction-type mechanism is granted. Paper [6] deals with a Multi-agent minimum-cost flow problem. It shows how Nash equilibria can be characterized by means of augmenting or decreasing path in a reduced network, also describes finding such equilibria that maximizes the flow value.

In thesis, we extend work published in [7]. Activities in the project are assigned to a particular agent. Agents are responsible for the duration of each activity, they can shorten the duration of their activities, incurring a crashing cost. If the project is finished before the normal value, agents get a bonus. Each agent wants to maximize its profit. The goal of the paper is to analyze the problem and to characterize stable strategy (Nash equilibria), i.e. in which no agent has an interest in modifying its own strategy if no strategy of another agent is changed. This model of the non-cooperative game differs from [8], where agents have no control over the duration of their activities and the situation is analyzed from a cooperative game perspective. Also, penalties are set up as difference from actual duration to the original estimate. In [7] penalties and rewards are fixed and pre-established. Formulation of a mixed integer linear programming model to find a stable strategy that minimizes the project makespan can be found in [9]. This paper extends the model from [7] such that the manager of the project can decide how to share the reward among the agents. Finding base strategy - strategy where every agent has non negative profit and the total project makespan is minimum, best Nash solution - every agent has a maximum profit and the total project

makespan is minimum, worst Nash solution - every agent has a maximum profit and the total project makespan is maximum and their comparison (Price of stability and Price of anarchy) are explained in paper [10].

1.4 Contribution

In this thesis, we extend the problem defined in [7] by definition of project milestones. We assume project activities are splitted among agents. Agents have an option to decrease the normal duration of activities to the incompressible limit. Project owner sets due dates and penalties for milestones in the project. Under these settings, the key contribution is the MILP formulation to determine the minimum project makespan among all stable solutions. The MILP effectively generates and adds lazy constraints to program. The experimental results show that the MILP model performs well on different types of testing data sets. Also, the price of anarchy and the price of stability are measured and discussed to get useful insight for the project manager.

1.5 Outline

The plan of the thesis is as follows. Section 2 formally defines the problem addressed in this thesis, along with the definition of the price of anarchy and the price of stability. In Section 3, the MILP formulation to compute the stable solution with minimum makespan using lazy constraint generation is provided. The experimental results are discussed in Section 4, and the last section concludes this work.

Chapter 2

Problem statement

2.1 Multi-agent project scheduling

In this section, we review the main definitions and concepts from [7] and extend the model by the definition of milestones. We assume basic knowledge of classical, single-agent time/cost trade-off problem. [11]

The MAPS problem with milestones is defined by a tuple $\langle G, \mathcal{A}, \underline{P}, \overline{P}, C, N_m \rangle$:

- **Project network.** $G = (N, E)$ is an activity-on-arc network. N is the set of nodes ($N = \{1, \ldots, n\}$) and E is the set of arcs. Arcs correspond to project activities, nodes represent events which allow specifying finish-start precedence relations. Arc of activities outgoing from a node cannot start before all incoming activities are finished. Specifying precedence relations can require the addition of dummy activities (arcs) having a zero duration in the project network [12]. Nodes 1 and n represent the project beginning and end, respectively.

- **Agents and strategies.** $\mathcal{A} = \{A_1, \ldots, A_a\}$ is the set of a agents. The set of activities belonging to agent A_u is denoted by E_u, with $E_u \cap E_v = \emptyset$, for $u \neq v$, and $\cup_{u=1}^{m} E_u = E$. The individual strategy of agent A_u, denoted by S_u, consists in deciding the duration of each activity $(i, j) \in E_u$, denoted by p_{ij}. We assume that activities duration are integer values. For any activity $(i, j) \in E$, we have: $\underline{p}_{ij} \leq p_{ij} \leq \overline{p}_{ij}$. Strategy S is a collection of a individual strategies S_u. Here we assume that an agent has total control over its activities only. Given strategy S, the project makespan is the length of the longest path from node 1 to node n on G, and is denoted by $D(S)$. To highlight the individual strategy of agent A_u within a certain strategy S, we sometime write S as (S_u, S_{-u}).

- **Crashing costs.** We let c_{ij} be the unit crashing cost of activity (i, j). So, the cost to shorten the duration of (i, j) from $\overline{p}_{ij}$ to p_{ij} is equal to $c_{ij}(\overline{p}_{ij} - p_{ij})$, and the cost paid by agent A_u for the strategy S_u is $\sum_{(i,j) \in E_u} c_{ij}(\overline{p}_{ij} - p_{ij})$.

- **Milestones and penalty.** N_m is a set of m milestones, $N_m \subseteq N$. For each milestone m, due date d^m is defined, having the following meaning. If the project reaches milestone m within d^m (i.e., if $t_m \leq d^m$), agent A_u pays no penalty. Otherwise, the agent A_u bears a penalty $q_{mu}T_m$, where $T_m = t_m - d^m$. Value q_{mu} represents a unit penalty cost paid by agent A_u for each unit time extension with respect to d^m of each milestone m.

- **Agent penalty.** Given a strategy S, the expense of agent A_u, denoted by $Z_u(S)$ is given by $Z_u(S) = \sum_{m \in M} q_{mu} \max\{0, t_m - d^m\} + \sum_{(i,j) \in E_u} c_{ij}(\overline{p}_{ij} - p_{ij})$.

The reaction of agent A_u to strategy S can be described by the following mathematical model:

$$\min \sum_{m \in N_m} q_{mu}T_m + \sum_{(i,j) \in \tau_u} c_{ij}(\overline{p}_{ij} - p_{ij})$$

s.t.

$$t_j - t_i - p_{ij} - s_{ij} = 0 \qquad \forall (i,j) \in E \qquad (2.1)$$

$$\underline{p}_{ij} \leq p_{ij} \leq \overline{p}_{ij} \qquad \forall (i,j) \in \tau_u \qquad (2.2)$$

$$p_{ij} = p_{ij}(S) \qquad \forall (i,j) \notin \tau_u \qquad (2.3)$$

$$t_m - d_m \leq T_m \qquad \forall m \in N_m \qquad (2.4)$$

where

$$q_{mu}, d_m, t_i, T_m, s_{ij} \in \mathbb{R}_{\geq 0}, \;\; p_{ij} \in \mathbb{Z}_{\geq 0}$$

Agent A_u is minimizing the penalty for tardy milestones $\sum_{m \in N_m} q_{mu}T_m$ and cost for shortening his activities $\sum_{(i,j) \in \tau_u} c_{ij}(\overline{p}_{ij} - p_{ij})$. Expression (2.1) defines precedence constraints given by G. Variable t_j represents the time, when node j is accomplished, or in other words when all activities coming to j are finished. Time t_j is equal to time t_i plus processing time of activity p_{ij} plus slack variable s_{ij}. Agent A_u can decrease processing time of activity (i,j) from normal duration $\overline{p}_{ij}$ to incompressible limit $\underline{p}_{ij}$ in activities he owns. This is expressed via constraints (2.2). Processing time of other agents activities are defined in strategy profile S given by constraints (2.3). Expression (2.4) sets up tardiness $T_m = \max\{0, t_m - d^m\}$, which is used in the objective function to compute the penalty for tardy milestones.

(*Critical graph* $\mathcal{G}(S)$) Given a strategy S, the longest paths from node 1 to every milestone m on G are called the critical paths. The arcs of the critical paths are critical activities. We call critical graph the subgraph $\mathcal{G}(S) \subseteq G(S)$ consisting of all critical activities.

2.2 Nash equilibria

In this section, we briefly review the characterization of Nash equilibria and poor strategy from [7] and provide definitions of increasing / decreasing cuts in a graph concerning the extension of the model.

(*Nash equilibria*) A strategy $\mathcal{S} = \{S_1, \ldots, S_a\}$ is a Nash equilibria if for all agents A_u and each strategy $S'_u \neq S_u$:

$$Z_u(S_{-u}, S_u) \geq Z_u(S_{-u}, S'_u) \tag{2.5}$$

Expression (2.5) means, that if S is a Nash equilibria, no agent A_u has the interest to change his strategy from S_u to S'_u, since all other agents do not change their strategies. Nash equilibria is a stable strategy.

(*Poor strategy*) A strategy $\mathcal{S} = \{S_1, \ldots, S_a\}$ with project duration $D(S)$ is a poor strategy if and only if there exists agent A_u and alternative strategy S'_u such that $Z_u(S) < Z_u(S'_u)$ and $D(S') = D(S)$, with $S' = (S_{-u}, S'_u)$

Vector S is a poor strategy if there exists at least one agent A_u, who can decrease his expenses by modifying processing times of his activities without affecting the project makespan. A strategy which is not poor is called non-poor. Nash equilibria is always a non-poor strategy.

Given non-poor strategy S with duration $D(S)$ agent A_u tries to find better strategy S', where his expenses are lower. Since S is non-poor, S' will increase or decrease the project makespan. Adjustment to S' can be characterized in terms of cuts in the critical graph $\mathcal{G}(S)$.

(*Cut in* $\mathcal{G}(S)$) Given a partition $(X, N \setminus X)$ of the set of activities N such that $0 \in X$ and also there must exists at least one milestone m such that $m \in N \setminus X$, 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 $\mathcal{G}(S)$.

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

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

So, all forward activities in a decreasing cut must have non-crash duration. Since all paths in $\mathcal{G}(S)$ are critical, this definition means that, decreasing (by 1 day) the length of all activities in $\omega^+(X)$, we decrease by at least 1 day the makespan of all milestones which belong to $N \setminus X$. Notice that if the milestone representing the end of the whole project is in X, then the project makespan is not affected by such operation. Also, notice that we are not requiring that all backward activities have to be increased: those having normal duration *may* simply become non-critical as a consequence of the application of the cut. Moreover, milestones in X may be affected by the decrease.

However, some makespans of milestones may be decreased by more than one day. This happens if all the paths leading to a certain milestone have 2 (or more) arcs in $\omega^+(X)$, and all such paths have at least one arc belonging to $\omega^-(X)$ which cannot be increased. Notice that this can happen only if certain arcs $(u, v) \in \omega^+(X)$ are such that no path fully contained in X exists from the initial node 0 to u. Call $\tilde{U} \subseteq X$ such set of nodes.

We can associate with each decreasing cut $\omega(X)$ a unit cost, consisting of the total cost of activities which are crashed, minus the total saving of extended activities, minus the total reward due to the reduction of the makespans of the milestones in $N \setminus X$:

$$W(X) = \sum_{(i,j)\in\omega^+(X)} c_{ij} - \sum_{(i,j)\in\omega^-(X)} c_{ij} - \sum_{m\in N\setminus X} q_m. \tag{2.6}$$

In general, (2.6) overestimates the real cost. As we already observed, a unit decrease in the activities of $\omega^+(X)$ may result in a multiple decrease of some milestone makespan(s). Call $\tilde{M}$ the set of milestones for which a multiple makespan reduction is achieved. Moreover, notice that $\tilde{U} \subseteq X$ may itself contain some milestone(s), for which the corresponding makespan is reduced: these are not accounted for in (2.6). So, if we let $\tilde{W}(X)$ be the real cost borne for reducing the activities of $\omega^+(X)$ (and extending all the activities of $\omega^-(X)$ which can be extended), it may happen that $\tilde{W}(X) < W(X)$.

Nonetheless, we can show that if X^* is a decreasing cut that minimizes $W(X)$, then $W(X^*) = \tilde{W}(X^*)$. Consider a decreasing cut $\omega(X)$ such that $W(X) > \tilde{W}(X)$. From the above discussion, this means that there is a node set $\tilde{U} \subseteq X$ of nodes such that, even if they lay in X, they are not reachable from the initial node through a path fully contained in X. Let us, therefore, consider the node subset $\tilde{X}$ obtained removing from X all the nodes of $\tilde{U}$, and consider a new cut $\omega(\tilde{X}) = \omega(X \setminus \tilde{U})$. Now, observe that in $\omega^+(\tilde{X})$ there can be no arc (u, v) such that $v \in \tilde{U}$, since otherwise the makespan of some milestone in $\tilde{M}$ would not be multiple reduced by the decreasing cut $\omega(X)$. As a consequence, passing from $\omega(X)$ to $\omega(\tilde{X})$, the set of forward arcs has not been enlarged, and no milestone on the right-hand side of $\omega(\tilde{X})$ experiences multiple reductions. So now the set $N \setminus \tilde{X}$ includes exactly all the milestones affected by the cut, and in conclusion, $\tilde{W}(\tilde{X}) = W(\tilde{X}) < W(X)$. Hence, given any cut $\omega(X)$, if $\tilde{W}(X) < W(X)$, there is certainly another cut $\omega(\tilde{X})$ such that $\tilde{W}(\tilde{X}) = W(\tilde{X}) < W(X)$, so indeed minimizing $W(X)$ is equivalent to minimizing $\tilde{W}(X)$.

(*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 idea is that if at least one forward activity of the cut is not at normal duration, we can get a saving from its extension. This is paid in terms of penalty for tardy milestones (in $N \setminus X$) which are extended. So, one can again associate with an increasing cut the following quantity, which is formally equivalent to Equation (2.6):

$$W(X) = \sum_{(i,j)\in\omega^+(X)} c_{ij} - \sum_{(i,j)\in\omega^-(X)} c_{ij} - \sum_{m\in N\setminus X} q_m. \tag{2.7}$$

This quantity is related to the saving achieved through the increase (by 1 day) of all forward activities which are not at normal duration (by definition there is at least one), the decrease of all backward activities and the fact that we pay penalty for tardy milestones $N\setminus X$.

Symmetrically to Equation (2.6), it may happen that (2.7) underestimates the real saving. The definition of increasing cut only requires that at least one forward activity (call it (u, v)) can be increased. The set $\omega^+(X)$ may contain other activities, but these may be at normal duration, so they cannot be indeed increased (this is taken care by letting $c_{ij} = 0$ in the residual graph). In conclusion, the increase of activity (u, v) may not affect the makespans of all milestones in $N \setminus X$. So, if we call $\tilde{W}(X)$ the real saving achieved by extending the activities of $\omega^+(X)$ (and reducing all the activities of $\omega^-(X)$ which can be reduced), it may happen that $\tilde{W}(X) > W(X)$.

Nonetheless, we can show that if X^* is an increasing cut that maximizes $W(X)$, then $W(X^*) = \tilde{W}(X^*)$. Suppose in fact that we have an increasing cut $\omega(X)$ and $W(X) > \tilde{W}(X)$. This means that there is a set $\tilde{M}$ of milestones such that, even if they lay in $N \setminus X$, their respective makespans are not increased by acting on the cut. Let us, therefore, consider the node subset $\tilde{N} \subset N \setminus X$ including all the nodes of $N \setminus X$ that are on some critical path leading to some node of $\tilde{M}$. Consider a new cut $\omega(\tilde{X}) = \omega(X \cup \tilde{N})$. Now, observe that in $\omega^-(\tilde{X})$ there can be no arc (u, v) such that $v \in \tilde{N}$, since otherwise the makespan of some milestone in $\tilde{M}$ would be affected by the increasing cut $\omega(X)$. As a consequence, passing from $\omega(X)$ to $\omega(\tilde{X})$, the set of forward arcs may have been enlarged (there may well be some "new" forward arc), but the set of affected milestones on the right-hand side of $\omega(\tilde{X})$ is smaller. Now the right-hand side includes exactly all the milestones affected by the cut, so in conclusion, $\tilde{W}(\tilde{X}) = W(\tilde{X}) > W(X)$. Hence, given any cut $\omega(X)$, if $\tilde{W}(X) > W(X)$, there is certainly another cut $\omega(\tilde{X})$ such that $\tilde{W}(\tilde{X}) = W(\tilde{X}) > W(X)$, so indeed minimizing $W(X)$ is equivalent to minimizing $\tilde{W}(X)$.

Notation	Meaning/description
$G = (N, E)$	Project network with N nodes/events and E arcs/activities
A_u	Agent
S_u	Strategy of agent A_u
S	Strategy profile, union of all agents strategies
p_{ij}	Processing time of activity $(i, j) \in E$
$\underline{p}_{ij}, \overline{p}_{ij}$	Incompressible limit and normal duration of activity $(i, j) \in E$
c_{ij}	Crashing cost of activity $(i, j) \in E$
t_i	Time, when node i is finished i
N_m	Set of milestones, $N_m \subseteq N$
d^m	Duedate of milestone m
q_{mu}	One day penalty for milestone m for agent A_u
T_m	Tardiness, $T_m = \max\{0, t_m - d^m\}$
$Z_u(S)$	Expenses of agent A_u under strategy profile S
$\mathcal{G}(S)$	Critical graph under strategy profile S
$D(S)$	Project makespan under strategy profile S
$\omega(X)$	Cut in graph $\mathcal{G}(S)$
$\omega^+(X), \omega^-(X)$	Decreasing and increasing cut in graph $\mathcal{G}(S)$
$W(X)$	Cut cost

Table 2.2: Notation

2.3 Illustrative example

Let us look on Figure 2.1. We consider project network $G = (N, E)$, where $N = \{1, \ldots, 4\}$ and $E = \{(1,2), (1,3), (2,3), (2,4), (3,4)\}$. Activities $(1,2), (2,3), (2,4) \in E_1$ and $(1,3), (3,4) \in E_2$. Project owner sets up 2 milestones, nodes 3 and 4. Milestone 3 has due date $d_3 = 5$, penalty $q_{3,1} = 120$ and $q_{3,2} = 120$. Milestone 4 has due date $d_4 = 7$, penalty $q_{4,1} = 10$ and $q_{4,2} = 190$.

Strategy $S = (5, 6, 2, 4, 2)$ is having all activities on the normal duration and project makespan $D(S) = 9$. With the strategy profile $S' = (4, 6, 2, 4, 2)$ in Figure 2.2, the makespan becomes 8, $Z_1(S') = 260$ and $Z_2(S') = 310$. This is a poor solution, agent A_1 can decrease his expenses by reducing the processing time of activity $(2, 3)$ to $p_{2,3} = 1$. Also, agent A_2 will reduce time of activity $(1, 3)$ to $p_{1,3} = 5$. This new strategy $S'' = (4, 5, 1, 4, 2)$ has $D(S'') = 8$, $Z_1(S'') = 230$ and $Z_2(S'') = 300$. No agent can change his processing time to decrease his expenses, so it is a non-poor strategy. Also, it is not possible to find another one with shorten makespan, so strategy $S'' = (4, 5, 1, 4, 2)$ is a Nash equilibria with the minimal makespan.

Changing penalty on milestone 4: $q_{4,1} = 200$ will force agent A_1 to finish milestone 4 on time. The optimal solution is $S''' = (3, 5, 2, 4, 2)$, having makespan $D(S''') = 7$. See Figure 2.3.

Changing the due date on milestone 3: $d_3 = 6$ will change the behavior of agents as well. The optimal solution is $S'''' = (5, 6, 1, 4, 2)$, having makespan $D(S'''') = 9$. See Figure 2.4. As we can see, agent A_1 reduces activity (2,3) to 1. This does not change the project makespan but will reduce expenses of agent A_1 from $Z_1(S) = 260$ to $Z_1(S'''') = 110$.

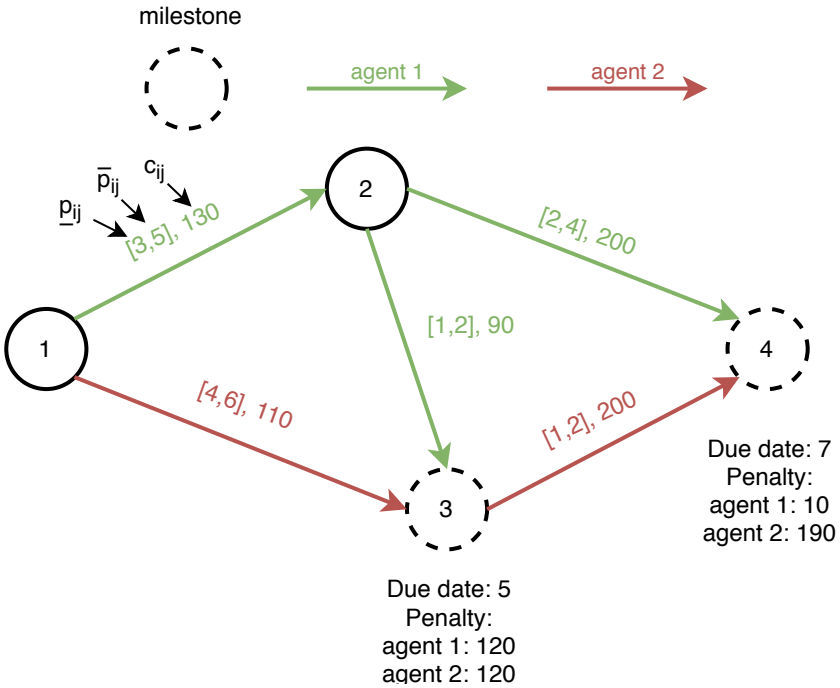


Figure 2.1: A multi-agent project network with 2 agents and 5 activities - instance

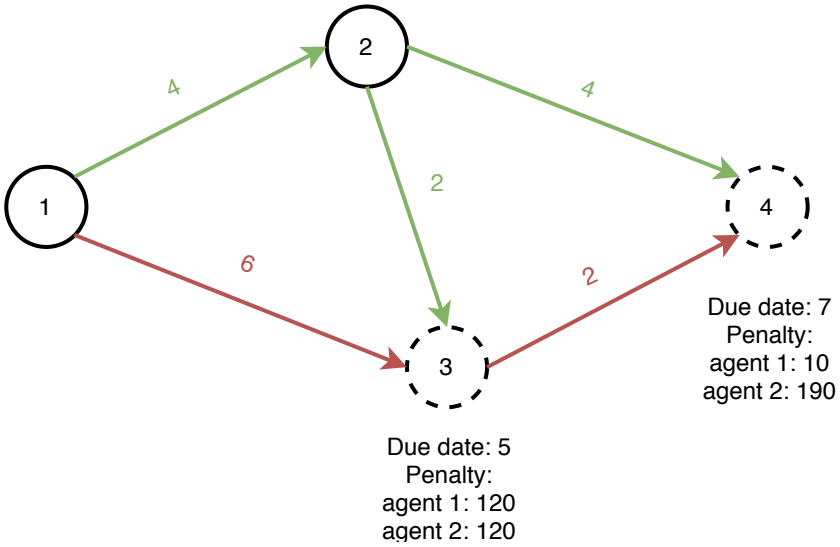


Figure 2.2: A multi-agent project network with 2 agents and 5 activities - strategy profile S'

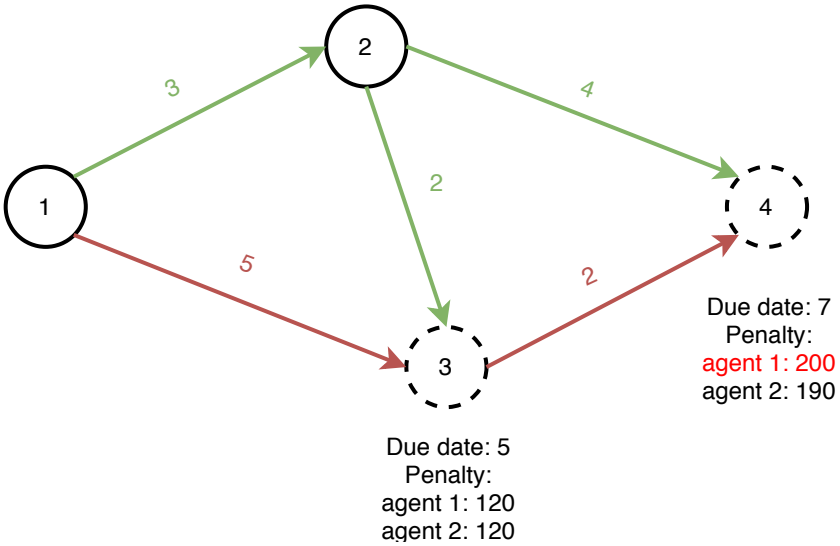


Figure 2.3: A multi-agent project network with 2 agents and 5 activities - strategy profile S'''

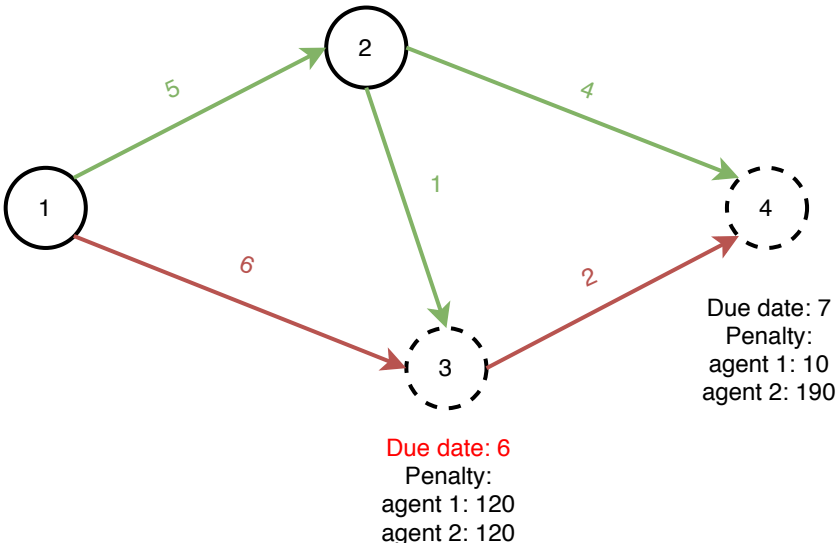


Figure 2.4: A multi-agent project network with 2 agents and 5 activities - strategy profile S''''

2.4 Price of anarchy and price of stability

In this section, we briefly review the characterization of PoA, PoS and base strategy from [10].

We focus on finding the Nash strategy which minimizes the project makespan S_{best}. Also, we can consider the Nash strategy which maximizes the project makespan S_{worst}. For the project owner, S_{best} and S_{worst} represent the best and the worst stable solutions. To assess the quality of strategies S_{best} and S_{worst}, we compare these Nash equilibria with base strategy S^*

(*Base strategy*) Given a tuple $\langle G, \mathcal{A}, \underline{P}, \overline{P}, C, N_m \rangle$, the base strategy is the strategy S^* having the minimum makespan among all strategies that guarantee all milestones are finished on time, and $D(S^*)$ denotes its makespan.

A base strategy can be computed by solving the following MILP:

$$
\begin{aligned}
&\min t_n && && (2.8)\\
&\text{s.t.} \\
&\quad t_j - t_i - p_{ij} - s_{ij} = 0 && \forall (i,j) \in E && (2.9)\\
&\quad t_0 = 0 && && (2.10)\\
&\quad \underline{p}_{ij} \le p_{ij} \le \overline{p}_{ij} && \forall (i,j) \in E && (2.11)\\
&\quad t_m = d_m && \forall m \in N_m && (2.12)\\
&\text{where} \\
&\ t_i, p_{ij} \in \mathbb{Z}_{\ge 0}, \ \ s_{ij} \in \mathbb{R}_{\ge 0}
\end{aligned}
$$

(*Price of anarchy and price of stability*) Given a tuple $\langle G, \mathcal{A}, \underline{P}, \overline{P}, C, N_m \rangle$ and the corresponding strategies S_{best}, S_{worst} and S^*, the ratios

$$PoA = \frac{D(S_{worst})}{S^*}$$

and

$$PoS = \frac{D(S_{best})}{S^*}$$

are called PoA and PoS respectively.

Chapter 3

MILP formulation

3.1 Main problem

In this chapter, we show how the MAPS can be solved as MILP with lazy constraints generation. First, we describe a formal MILP formulation of the MAPS with milestones problem:

$$\min \left(t_n + \frac{\sum_{\forall (i,j)\in E} c_{ij}\left(\overline{p}_{ij} - p_{ij}\right) + \sum_{\forall m \in N_m} \sum_{\forall u \in A} q_{mu} T_m}{1 + \sum_{\forall (i,j)\in E} c_{ij}\left(\overline{p}_{ij} - \underline{p}_{ij}\right) + \sum_{\forall m \in N_m} \sum_{\forall u \in A} q_{mu}(t_m^{max} - d_m)} \right) \tag{3.1}$$

s.t.

$$t_j - t_i - p_{ij} - s_{ij} = 0 \quad \forall (i,j) \in E \tag{3.2}$$

$$t_0 = 0 \tag{3.3}$$

$$\underline{p}_{ij} \leq p_{ij} \leq \overline{p}_{ij} \quad \forall (i,j) \in E \tag{3.4}$$

$$W(\omega_{dec}^{(u)}) < 0 \quad \forall \omega_{dec}^{(u)}, \forall A_u \in A \tag{3.5}$$

$$W(\omega_{inc}^{(u)}) > 0 \quad \forall \omega_{inc}^{(u)}, \forall A_u \in A \tag{3.6}$$

$$t_m - d_m \leq T_m \quad \forall m \in N_m \tag{3.7}$$

where

$$t_i, T_m, p_{ij} \in \mathbb{Z}_{\geq 0}, \;\; s_{ij} \in \mathbb{R}_{\geq 0}$$

The objective function (3.1) has two parts. Th first one minimizes project makespan t_n, the second part is to ensure the optimal solution is non-poor. As t_n is an integer value and the second term is always smaller than 1, it will not affect the project makespan. Expression (3.2) defines precedence constraints, and (3.3) will force the model to start at time 0. Processing time of activity is lower or equal than normal duration and higher or equal than the incompressible limit, due to constraints (3.4). Constraints (3.5) and (3.6) defined in Section 2.2 guarantee there is no profitable decreasing or increasing cut. Expression (3.7) sets up the tardiness T_m, which is used in the objective function.

In order to express constraints (3.5) and (3.6) in linear term, we define residual network $N_u(S)$ and new decision variables $x_{i,j}, y_{i,j}, z_{i,j}, z_m, z'_m$.

3.2 Residual graph

Residual network $N_u(S)$ is a function of strategy S. We introduce new binary variables x_{ij},y_{ij},z_{ij},z_m,z'_m. Variable x_{ij} is equal to 1, iff processing time p_{ij} can be increased. Variable y_{ij} is equal to 1, iff processing time p_{ij} can be decreased. Variable z_{ij} is equal to 1, iff activity (i,j) is on the critical path in the critical graph $\mathcal{G}(S)$. Variable z_m is equal to 1, iff $t_m - d_m \geq 0$ for milestone m and z'_m is equal to 1, iff $t_m - d_m > 0$ for milestone m. We append the following constraints to the previous MILP formulation:

$$\epsilon - z_{ij} \leq s_{ij} \leq \overline{s}_{ij}\,(1 - z_{ij}) \qquad \forall (i,j) \in E \qquad (3.8)$$

$$z_{ij} \leq \sum_{\forall (k,i) \in E} z_{ki} \qquad \forall (i,j) \in E : i > 1 \qquad (3.9)$$

$$z_{ij} \leq \sum_{\forall (j,l) \in E} z_{jl} \qquad \forall (i,j) \in E : j < n, j \notin N_m \qquad (3.10)$$

$$\sum_{\forall (i,m) \in E} z_{im} \geq 1 \qquad \forall m \in N_m \qquad (3.11)$$

$$1 \leq \sum_{\forall (1,i) \in E} z_{1i} \qquad (3.12)$$

$$t_m \leq d_m - 1 + (t_m^{max} - t_m^{min} + 1) z_m \qquad \forall m \in N_m \qquad (3.13)$$

$$t_m \geq d_m - (t_m^{max} - t_m^{min})(1 - z_m) \qquad \forall m \in N_m \qquad (3.14)$$

$$t_m \leq d_m + (t_m^{max} - t_m^{min}) z'_m \qquad \forall m \in N_m \qquad (3.15)$$

$$t_m \geq d_m + 1 - (t_m^{max} - t_m^{min} + 1)(1 - z'_m) \qquad \forall m \in N_m \qquad (3.16)$$

$$x_{ij} \leq \left(\overline{p}_{ij} - p_{ij}\right) \leq \left(\overline{p}_{ij} - \underline{p}_{ij}\right) x_{ij} \qquad \forall (i,j) \in E \qquad (3.17)$$

$$y_{ij} \leq \left(p_{ij} - \underline{p}_{ij}\right) \leq \left(\overline{p}_{ij} - \underline{p}_{ij}\right) y_{ij} \qquad \forall (i,j) \in E \qquad (3.18)$$

$$z_{ij} \geq x_{ij} \qquad \forall (i,j) \in E \qquad (3.19)$$

Expression (3.8) defines, that if the activity (i,j) has a zero slack variable s_{ij}, then it cannot be critical. Also, when the activity (i,j) is critical, it cannot have $s_{ij} > 0$.

Expressions (3.9) and (3.10) ensure continuity of critical paths in the graph. If an activity (i,j) is critical, then at least 1 activity entering node i is critical too, due to (3.9). Also if activity (i,j) is critical and node j is not a milestone, then at least one activity leaving node j is critical, due to (3.10). Constraints (3.11) and (3.12) defines that a critical path exists to every milestone and minimum one critical path is starting from the first node . Variables z_m, z'_m are used to determine tardy milestones via constraints (3.13) - (3.16). Expression (3.17) sets up x variable and (3.18) sets up y variable. If activity is not critical, it cannot be increased, due to (3.19)

For each agent A_u, we construct residual graph $N_u(S)$ from x_{ij}, y_{ij}, z_{ij}. The flow on arcs is bounded between l_{ij}^u and u_{ij}^u. Values of variables l_{ij}^u and u_{ij}^u are specified by the transformation shown in Table 3.3, for more information about the transformation please see [10].

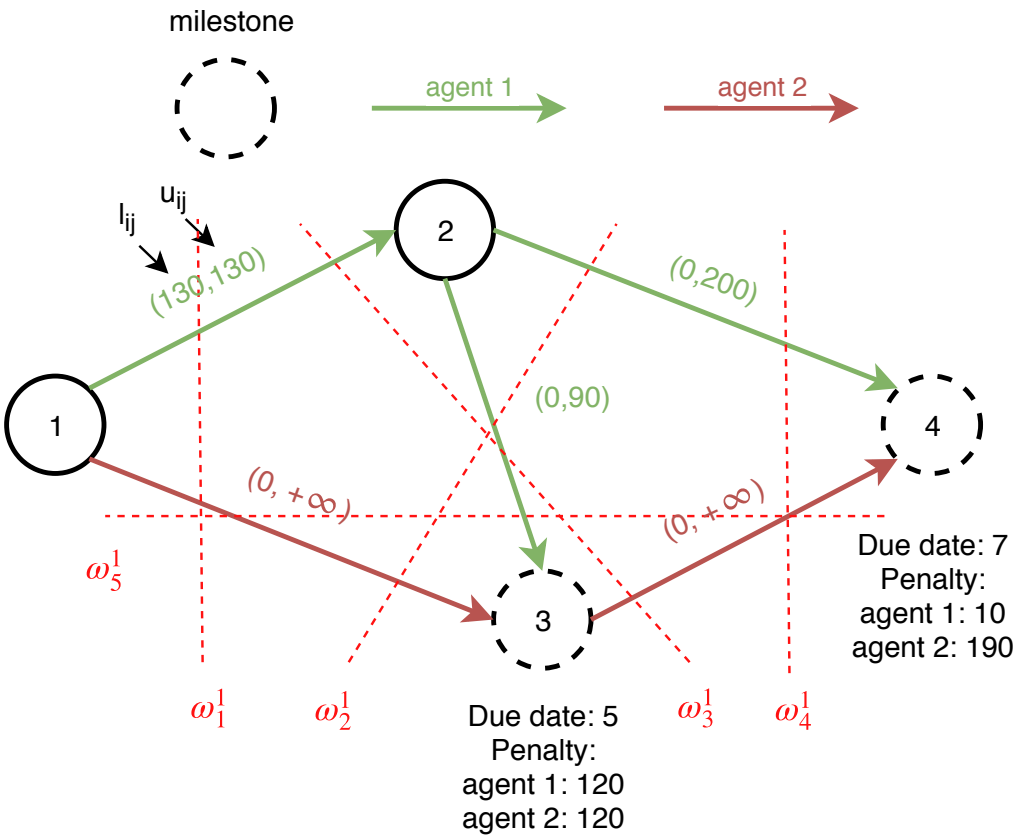
$(i,j) \in A_u$	z_{ij}	x_{ij}	y_{ij}	l^u_{ij}	u^u_{ij}
True	0	0/1	0/1	0	0
True	1	1	1	c_{ij}	c_{ij}
True	1	0	1	0	c_{ij}
True	1	1	0	c_{ij}	$+\infty$
True	1	0	0	0	$+\infty$
False	0	0/1	0/1	0	0
False	1	0/1	0/1	0	$+\infty$

Table 3.3: Lower and upper capacities l^u_{ij}, u^u_{ij} in $N_u(S)$

The most profitable decreasing or increasing cuts are computed independently in subproblems MILP. See examples of residual graphs and cuts for different strategies and agents in Figures 3.1 - 3.3.

In Figure 3.1, we can see an example of residual network $N_1(S')$ constructed for agent A_1 and strategy $S' = (4, 6, 2, 4, 2)$ from Figure 2.2. There are 5 possible cuts in the network, but only one is profitable: increasing cut ω^1_3, $W(\omega^1_3) = 30$. Agent $A_!$ will increase the duration of activity $p_{12} = 5$ and decrease the activity $p_{23} = 1$.

Figures 3.2 and 3.3 are the residual networks for both agents and strategy $S'' = (4, 5, 1, 4, 2)$. It is a non-poor strategy, and also there does not exist any profitable cut. It is an optimal solution.



Increasing cuts

W(ω_1^1) = 130 + 0 -120 -10 = 0

W(ω_2^1) = 0 + 0 + 0 - 120 - 10 = -130

W(ω_3^1) = 130 - 90 + 0 -10 = 30

W(ω_4^1) = 0 + 0 - 10 = -10

W(ω_5^1) = 0 - ∞ - 120 = -∞

Maximum cost: 30 > 0 =>
Profitable cut found

Decreasing cuts

W(ω_1^1) = 130 + ∞ -120 -10 = +∞

W(ω_2^1) = 200 + 90 + ∞ - 120 - 10 = +∞

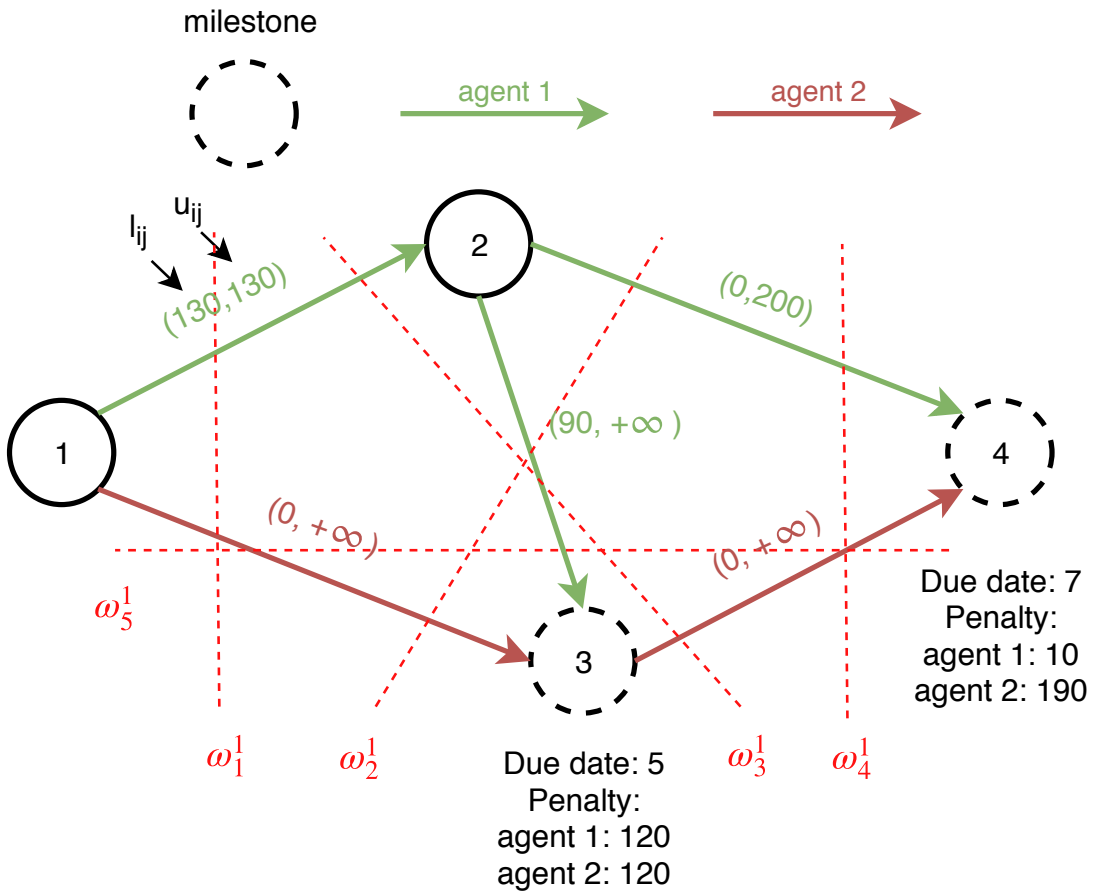
W(ω_3^1) = 130 - 0 + ∞ -10 = +∞

W(ω_4^1) = 200 +∞ - 10 = +∞

W(ω_5^1) = ∞ - 0 - 120 = +∞

Minimum cost: +∞ > 0 =>
No profitable cut

Figure 3.1: Residual network $N_1(S')$ for strategy profile S' from Figure 2.2 for agent A_1



Increasing cuts

$W(\omega_1^1) = 130 + 0 - 120 - 10 = 0$

$W(\omega_2^1) = 0 + 90 + 0 - 120 - 10 = -40$

$W(\omega_3^1) = 130 - \infty + 0 - 10 = -\infty$

$W(\omega_4^1) = 0 + 0 - 10 = -10$

$W(\omega_5^1) = 0 - \infty - 120 = -\infty$

Maximum cost: 0 = 0 =>
No profitable cut found

Decreasing cuts

$W(\omega_1^1) = 130 + \infty - 10 = +\infty$

$W(\omega_2^1) = 200 + \infty + \infty - 10 = +\infty$

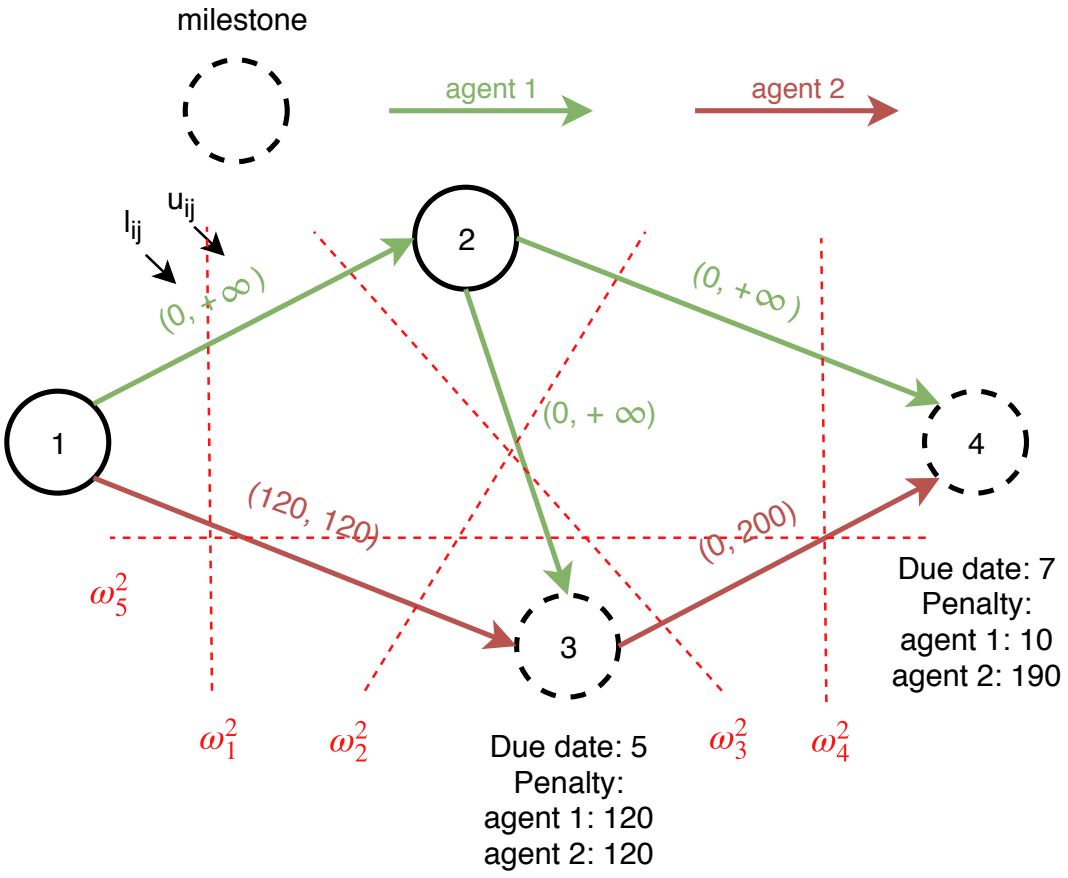
$W(\omega_3^1) = 130 - 90 + \infty - 10 = +\infty$

$W(\omega_4^1) = 200 + \infty - 10 = +\infty$

$W(\omega_5^1) = \infty - 0 = +\infty$

Minimum cost: $+\infty > 0$ =>
No profitable cut

Figure 3.2: Residual network $N_1(S'')$ for strategy profile S'' from Figure 2.3 for agent A_1



Increasing cuts

W(ω_1^2) = 120 + 0 -120 -190 = -190

W(ω_2^2) = 0 + 0 + 120 - 120 - 190 = -190

W(ω_3^2) = 0 - ∞ + 0 -190 = - ∞

W(ω_4^2) = 0 + 0 - 190 = -190

W(ω_5^2) = 120 - 200 - 120 = - 200

Maximum cost: 0 = 0 =>
No profitable cut found

Decreasing cuts

W(ω_1^2) = ∞ + 120 - 190 = +∞

W(ω_2^2) = ∞ + ∞ + 120 - 190 = +∞

W(ω_3^2) = ∞ - 0 + 200 -190 = +∞

W(ω_4^2) = ∞ + 200 - 190 = +∞

W(ω_5^2) = 120 - 0 = 120

Minimum cost: 120 > 0 =>
No profitable cut

Figure 3.3: Residual network $N_2(S'')$ for strategy profile S'' from Figure 2.3 for agent A_2

3.3 Subproblem

We compute profitable increasing and decreasing cuts separately in subproblem MILP. We use variables l_{ij} and u_{ij} from residual network $N_u(S)$, also variables z_m and z'_m from the main MILP formulation. After we find a profitable decreasing / increasing cut, we construct a constraint and add it lazy to the main MILP formulation. See Section 3.4 for more details about lazy constraint generation.

Decreasing Cuts

In this section, you can find a MILP formulation to find the most profitable decreasing cut in the residual network. We define $M^C = \{m \in N_m : z'_m = 1\}$, the set of tardy milestones, where variable z'_m is equal to 1, iff $t_m - d_m > 0$ for milestone m. M^C is set of all tardy milestones, which can be affected by a decreasing cut. The cut is profitable when the objective function is less than 0.

$$\min \sum_{(i,j)\in E} \alpha_{ij} u_{ij} - \sum_{(i,j)\in E} \beta_{ij} l_{ij} - \sum_{m\in M^C} q_{mu}(1-\gamma_m)$$

s.t.

$$\alpha_{ij} - \beta_{ij} = \gamma_i - \gamma_j \qquad \forall (i,j) \in E \qquad (3.20)$$

$$\alpha_{ij} + \beta_{ij} \leq 1 \qquad \forall (i,j) \in E \qquad (3.21)$$

$$\gamma_1 = 1 \qquad (3.22)$$

$$\textstyle\sum_{m\in N_m} \gamma_m \leq |N_m| - 1 \qquad (3.23)$$

where

$$q_{mu}, \ell_{ij}, u_{ij} \in \mathbb{R}_{\geq 0}, \;\; \alpha_{ij}, \beta_{ij}, \gamma_i \in \{0,1\}$$

We added binary variables $\alpha_{ij}, \beta_{ij}, \gamma_i$ to define a cut in residual network $N_u(S)$. Variable α_{ij} is equal to 1, iff arc (i,j) is a forward arc in the cut. Variable β_{ij} is equal to 1, iff arc (i,j) is a backward arc in the cut. Variable γ_i is equal to 1, iff node i is on the left in the cut, i.e. $i \in X$.

Constraints (3.20) means, that when activity (i,j) is forward arc, than node i is on the left and j is on the ride. But also, if the activity (i,j) is the backward arc, then node i is on the right and node j on the left. Arc (i,j) cannot be forward and backward at the same time, due to (3.21). The first node is always on the left side of cut due to constraints (3.22) and there is always at least one tardy milestone on the ride side of the cut (3.23).

If a profitable cut is found, we define paramteres of new lazy constraint $LC_k^{dec} = (F_k, B_k, C_k, D_k)$ where

$F_k = \{(i,j) \in E : \alpha_{ij} = 1 \wedge z_{i,j} = 1\}$,
$B_k = \{(i,j) \in E : \beta_{ij} = 1 \wedge z_{i,j} = 1\}$,
$C_k = \{(i,j) \in E : \alpha_{ij} = 1 \wedge z_{i,j} = 0\}$,
$D_k = \{m \in N_m : \gamma_m = 0 \wedge z'_m = 1\}$.

The corresponding constraint is added lazy to main MILP formulation, as will be explain later in Section 3.4.

Increasing Cuts

In this section, you can find a MILP formulation to find the most profitable increasing cut in the residual network. We define $M^C = \{m \in N_m : z_m = 1\}$, the set of tardy milestones. Variable z_m is equal to 1, iff $t_m - d_m >= 0$ for milestone m. The set M^C of tardy milestones are milestones, which can be affected by increasing cut. The set M^C differs for increasing and decreasing cut. In decreasing cut, we select milestones which are tardy, in increasing we select tardy milestones and also milestones where $t_m = d_m$. The Increasing cut will increase the makespan of milestones, so it will make milestones where $t_m = d_m$ tardy. The cut is profitable when the objective function is more than 0.

$$\max \sum_{(i,j)\in E} \alpha_{ij} l_{ij} - \sum_{(i,j)\in E} \beta_{ij} u_{ij} - \sum_{m \in M^C} q_{mu}(1 - \gamma_m)$$

s.t.

$$\alpha_{ij} - \beta_{ij} = \gamma_i - \gamma_j \qquad \forall (i,j) \in E \qquad (3.24)$$

$$\alpha_{ij} + \beta_{ij} \leq 1 \qquad \forall (i,j) \in E \qquad (3.25)$$

$$\gamma_1 = 1 \qquad (3.26)$$

$$\textstyle\sum_{m \in N_m} \gamma_m \leq |N_m| - 1 \qquad (3.27)$$

where

$$q_{mu}, \ell_{ij}, u_{ij} \in \mathbb{R}_{\geq 0}, \;\; \alpha_{ij}, \beta_{ij}, \gamma_i \in \{0,1\}$$

Definition of variables $\alpha_{ij}, \beta_{ij}, \gamma_i$ and contraints are identical with the decreasing cut.

If a profitable cut is found, we define parameters of new lazy constraint $LC_k^{inc} = (F_k, B_k, C_k, D_k)$ where

$F_k = \{(i,j) \in E : \alpha_{ij} = 1 \wedge z_{i,j} = 1\}$,
$B_k = \{(i,j) \in E : \beta_{ij} = 1 \wedge z_{i,j} = 1\}$,
$C_k = \{(i,j) \in E : \beta_{ij} = 1 \wedge z_{i,j} = 0\}$,
$D_k = \{m \in N_m : \gamma_m = 0 \wedge z_m = 0\}$.

The corresponding constraint is added lazy to main MILP formulation, as will be explain later in Section 3.4.

3.4 Lazy constraints generation

We have shown how to compute profitable decreasing and increasing cut for each agent in Section 3.3. We add these constraints lazy to main MILP model. Instead of constraint (3.5), we add:

$$\sum_{\forall(i,j)\in F_k} y_{ij} + \sum_{\forall(i,j)\in B_k} x_{ij} \leq |F_k \cup B_k| - 1 + \sum_{\forall(i,j)\in C_k} z_{ij} + \sum_{\forall m\in D_k} (1 - z'_m) \quad \forall LC_k^{dec} \in LC^{dec} \tag{3.28}$$

and instead of (3.6), we add:

$$\sum_{\forall(i,j)\in F_k} x_{ij} + \sum_{\forall(i,j)\in B_k} y_{ij} \leq |F_k \cup B_k| - 1 + \sum_{\forall(i,j)\in C_k} z_{ij} + \sum_{\forall m\in D_k} z_m \quad \forall LC_k^{inc} \in LC^{inc} \tag{3.29}$$

The sum of forward and backward arcs in the cut associated with variables $x_{i,j}$ and $y_{i,j}$ lower or equal to the union of sets -1 generates a constraint in the main MILP model. The constraint is added lazy to the model and forces the model to change from strategy S to a strategy S' where this cut will not exist (no such configuration of forward and backward arcs).

Blocking arcs and milestones

Let us look at Figure 3.4. Consider this situation: Forward arcs in an increasing cut are $\alpha_{2,5} = 1, \alpha_{3,6} = 1$ and $\alpha_{4,7} = 1$, backward arc is $\beta_{5,3} = 1$, critical arcs are $z_{2,5} = 1$ and $z_{3,6} = 1$, milestone 8 is tardy $z_8 = 1$ and milestone 9 is not tardy $z_9 = 0$. We find the profitable increasing cut $60 + 50 - 100 (> 0)$, and set up $F_k = \{(2,5),(3,6)\}, B_k = \emptyset, C_k = \{(5,3)\}, D_k = \{9\}$. This cut will generate a constraint: $x_{2,5} + x_{3,6} <= 1 + z_{5,3} + z_9$. Arc $(5,3)$ is called blocking, because we have to turn off the constraint, if this arc becomes critical ($z_{5,3} = 1$). The reason is when $(5,3)$ becomes critical, cut costs will change: $60 - 20 + 50 = -10 (< 0)$ and the cut is no more profitable. On the other hand, when activity $(4,7)$ becomes critical, the constraint can stay in model, because it only improves the cut cost $60 + 50 + 20 - 100 (> 0)$. Also the cut can becomes unprofitable, when milestone 9 turns critical $60 + 50 - 100 - 100 (< 0)$, and constrain has to be turned off by variable z_9.

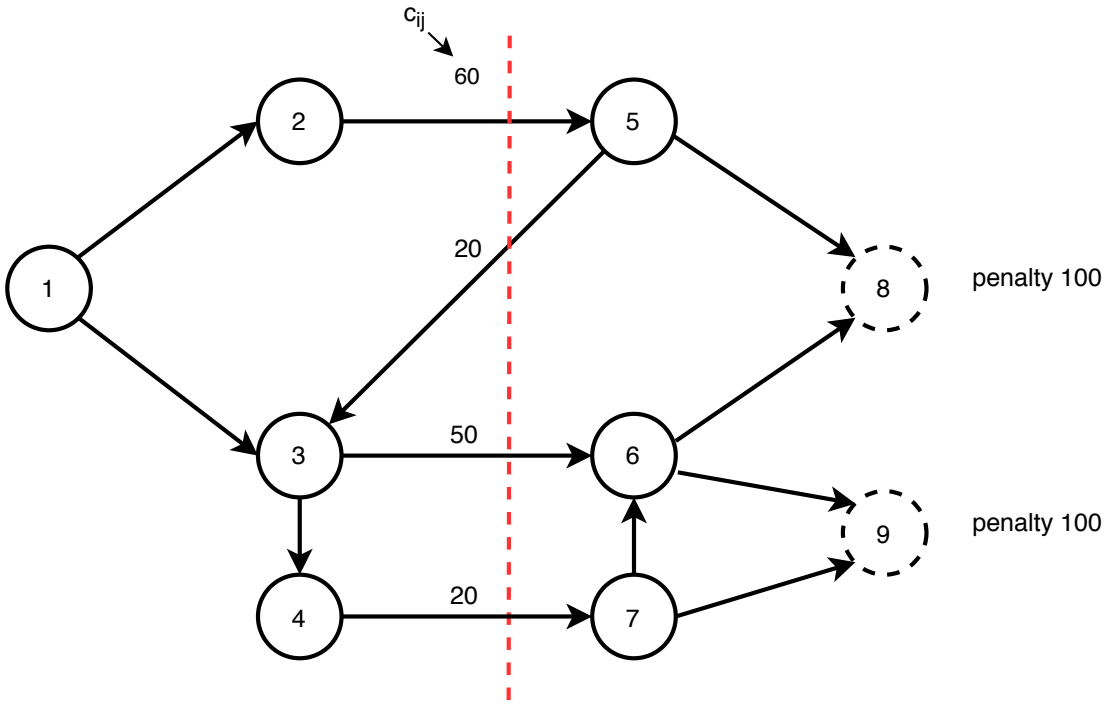


Figure 3.4: Blocking arcs and milestones for an increasing cut in the residual graph

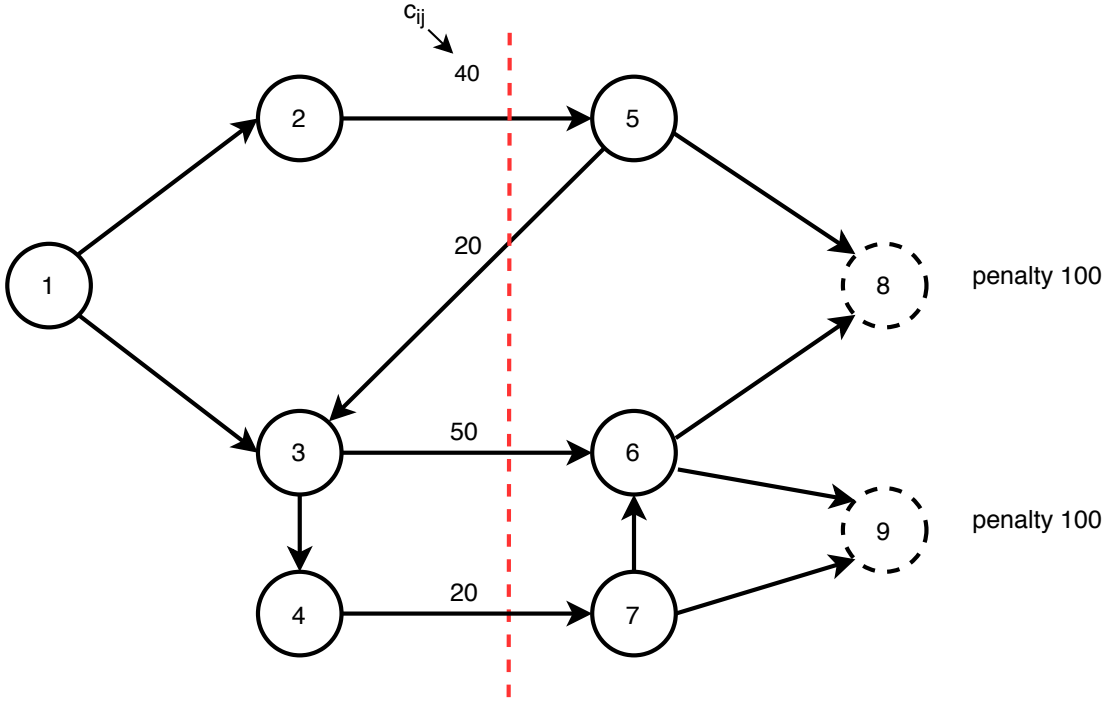


Figure 3.5: Blocking arcs and milestones for a decreasing cut in the residual graph

Figure 3.5 illustrates the way decreasing cuts are transformed to lazy contraints. Consider the situation where $\alpha_{2,5} = 1, \alpha_{3,6} = 1$ and $\alpha_{4,7} = 1$, backward arc is $\beta_{5,3} = 1$, critical arcs are $z_{2,5} = 1$ and $z_{3,6} = 1$, milestone 8 is tardy $z_8 = 1$ and milestone 9 is not tardy $z_9 = 0$. In this configuration, a decreasing profitable cut appears $40 + 50 - 100(< 0)$. Cut is defined as $F_k = \{(2,5),(3,6)\}, B_k = \emptyset, C_k = \{(4,7)\}, D_k = \{8\}$. It will generate constraint $x_{2,5} + x_{3,6} <= 1 + z_{4,7} + (1 - z'_8)$. Arc $(4,7)$ is blocking, if it becomes critical, we have to turn the constraint off, because the cut will no longer be profitable $40 + 20 + 50 - 100(> 0)$. The arc $(5,3)$ can only improve the cut cost $40 - 20 + 50 - 100(< 0)$, so it is not blocking. If milestone 8 is not tardy, the cut also has to be turned off, because the cost will not be profitable $40 + 50(> 0)$.

Chapter 4

Experimental results

The algorithm performance was measured on a personal computer with macOS Mojave version 10.14.4, 8 GB of RAM and 2,5 GHz Intel Core i5 processor. The MILP model was solved with Gurobi solver version 8.0.1. Section 4.1 explains how the instances were generated and provides step by step example of program execution. Section 4.2 discusses the performance of MILP model under different conditions: number of activities, number of agents, number of milestones and others. Price of anarchy and price of stability is addressed at the end of the section.

4.1 Benchmark instances

Problem instances were generated by RanGen1, generator defined in [13]. For each problem size, 100 instances were created. All of them were made with Order Strength equal to 0.3 which represents the number of precedence relations divided by the theoretical maximum of precedence relations in the network. Instances were converted from activity-on-node to activity-on-arcs by definition in [9], using an algorithm described in [12]. The converter was modified to generate milestones in the graph. Each milestone has a due date d_m and penalty for agents $q_{mu} = q_m w_{mu}$ assigned. The number of milestones in the graph is a parameter for a converter, a value in interval $[0, 1]$ is expected. It is a ration of the number of milestones to all nodes in the graph. The last node is always a milestone and the first one never is. The due date d_m is calculated in an interval of the longest path to milestone using lower bounds on the arcs and the longest path to milestone using upper bounds on the arcs. A parameter in interval $[0, 1]$ is expected, 0 means d_m will have the smallest possible value, 1 will compute the biggest value. The same principle is enforced in the calculation of penalties, were boundaries are calculated and parameter in interval $[0, 1]$ sets the final value of penalty q_{mu}.

Step by step example

Let us look step by step on program execution from the beginning to the end. We take the example from Section 2.3. Consider an input file:

```
NumNodes = 6;
NumAgents = 3;
Arcs = {
//fromnode,tonode,cost,pLB,pUB,agent;
<    1,    2,     0,    0,    0,    1>,
<    2,    3,   130,    3,    5,    2>,
<    2,    4,   110,    4,    6,    3>,
<    3,    4,    90,    1,    2,    2>,
<    3,    5,   200,    2,    4,    2>,
<    4,    5,   200,    1,    2,    3>,
<    5,    6,     0,    0,    0,    1>
};
Milestones = {
//m,dm,qm,wum;
<    6,    7,    200, [0.000000,0.1,0.9]>,
<    4,    5,    240, [0.000000,0.5,0.5]>
};
```

The input file can be visualized using Graphviz— Open Source Graph Drawing Tools described in [14], see Figure 4.1.

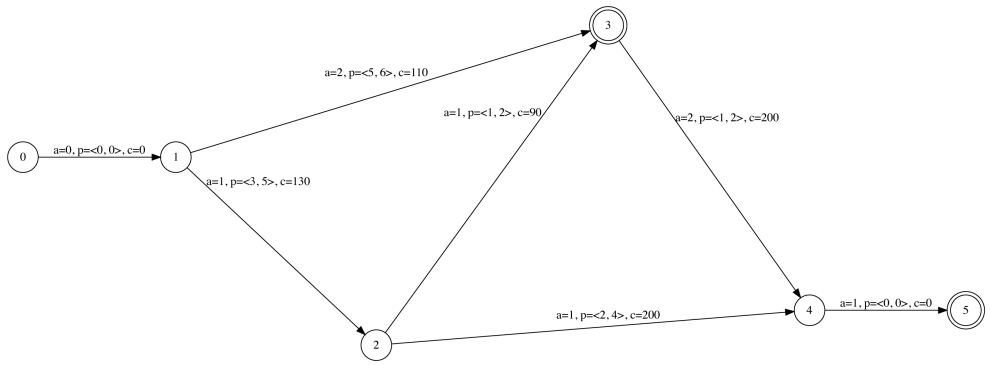


Figure 4.1: Step by step example - input file

We consider project network $G = (N, E)$, where $N = \{0, \ldots, 5\}$ and $E = \{(0,1), (1,2), (1,3), (2,3), (2,4), (3,4), (4,5)\}$. Activities $(1,2), (2,3), (2,4) \in E_1$ and $(1,3), (3,4) \in E_2$. Project owner sets up 2 milestones, nodes 3 and 5. Milestone 3 has due date $d_3 = 5$, penalty $q_{3,1} = 120$ and $q_{3,2} = 120$. Milestone 5 has due date $d_5 = 7$, penalty $q_{5,1} = 10$ and $q_{5,2} = 190$.

After the main MILP starts, the first iteration comes up with the strategy $S = (0, 3, 4, 1, 2, 1, 0)$, having makespan $D(S) = 5$. The A_1 agent has expenses $Z_1(S) = 750$ and agent A_2 has $Z_2(S) = 420$. No milestones are tardy. In Figure 4.2, you can see the project network in the MILP callback function. Critical arcs are bold. The Program makes the residual network for both agents respectively and finds two profitable increasing cuts. The first increasing cut w_1^1 is agent's A_1 and has a cost $W(w_1^1) = 290$. Agent A_1 can increase the activity on arcs $(2,3)$, $(2,4)$ and saves 290. The second increasing cut w_1^2 is agent's A_2 and has a cost $W(w_1^2) = 200$. Agent A_2 can increase the activity on arc $(3,4)$ and saves 200. The program chooses cut w_1^2 and generate a lazy constraint $x_{3,4} \leq 0 + z_0$.

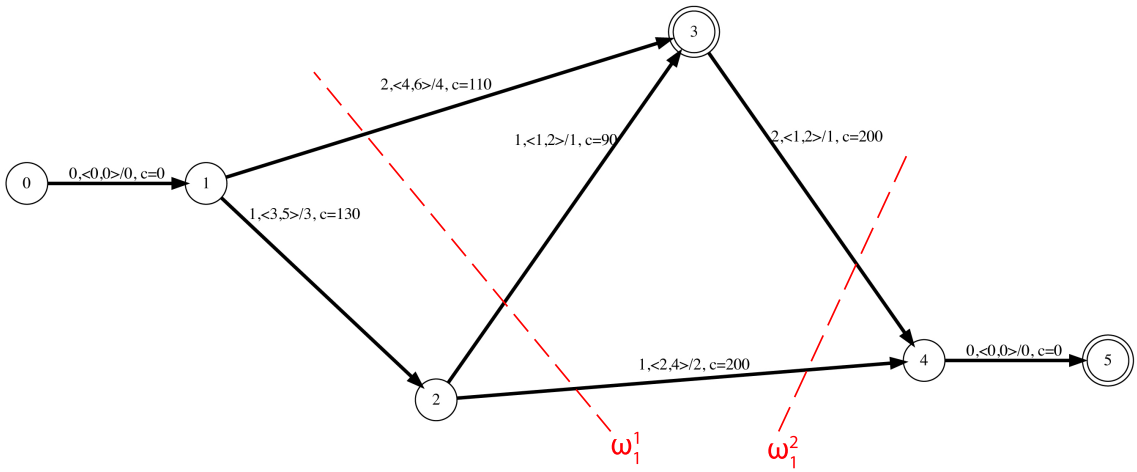


Figure 4.2: Step by step example - 1^{st} callback

Next iteration of algorithm brings the strategy $S^1 = (0, 3, 4, 1, 3, 2, 0)$, see Figure 4.3. The A_1 agent has expenses $Z_1(S^1) = 550$ and agent A_2 has $Z_2(S^1) = 220$. No milestones are tardy. In the MILP callback function, there are 2 profitable increasing cuts. The first one w_1^1 is the same as in Figure 4.2. The second increasing cut w_1^2 is agent's A_2 and has a cost $W(w_1^2) = 110$. Agent A_2 can increase the activity on arc $(1,3)$ and saves 110. The program chooses cut w_1^2 and generates a lazy constraint $x_{1,3} \leq 0 + z_0 + z_1$.

In Figure 4.4, you can see the strategy $S^2 = (0, 3, 5, 2, 4, 2, 0)$ evaluated in the next iteration of the algorithm. The A_1 agent has expenses $Z_1(S^2) = 260$ and agent A_2 has $Z_2(S^2) = 110$.

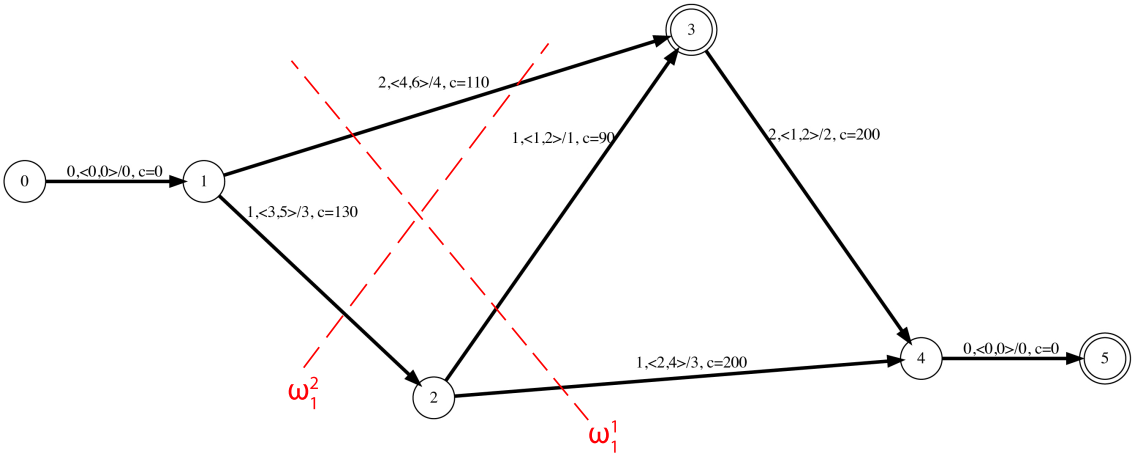


Figure 4.3: Step by step example - 2^{nd} callback

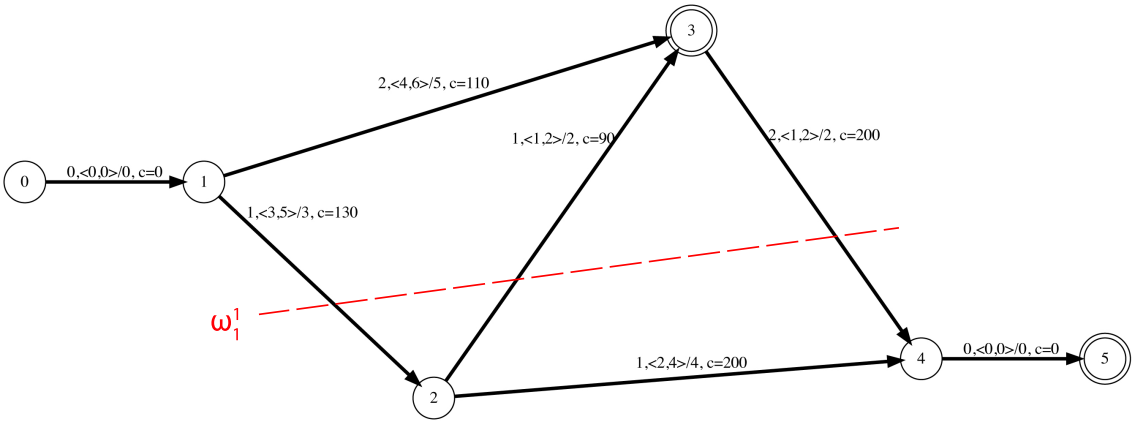


Figure 4.4: Step by step example - 3^{rd} callback

No milestones are tardy. One profitable increasing cut w_1^1 is found in the MILP callback function. Agent A_1 can decrease his expenses when he increases the duration of activity $(1, 2)$ and decreases the duration of activity $(2, 3)$. The cut cost is $W(w_1^1) = 40$. This cut will generate a constraint $x_{1,2} + y_{2,3} \leq 1$.

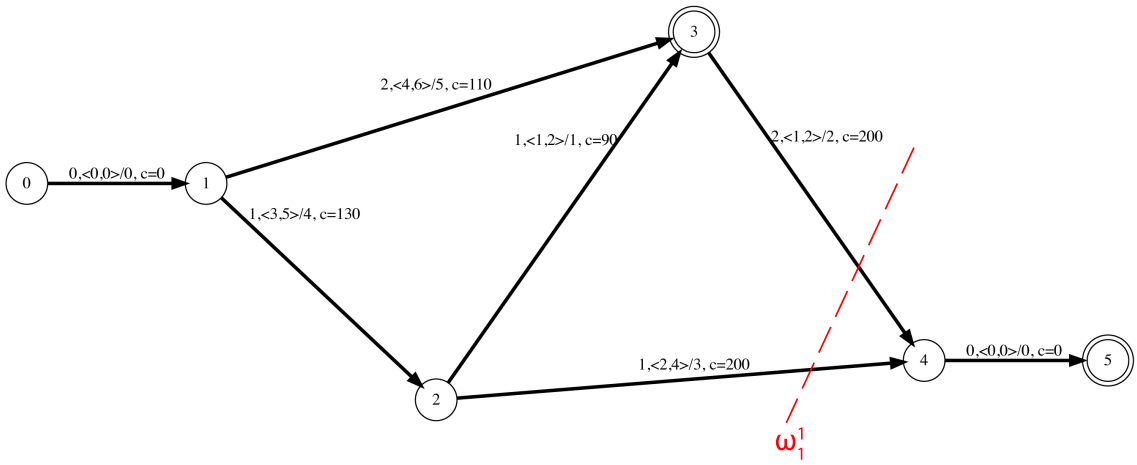


Figure 4.5: Step by step example - 4^{th} callback

In the next iteration of algorithm, the strategy is $S^3 = (0, 4, 5, 1, 3, 2, 0)$, see Figure 4.5. The A_1 agent has expenses $Z_1(S^3) = 420$ and agent A_2 has $Z_2(S^3) = 110$. No milestones are tardy. One profitable increasing cut w_1^1 is found in the MILP callback function. Agent A_1 can decrease his expenses when he increases the duration of activity $(2, 4)$, even though he pays the penalty because milestone 5 becomes tardy. The cut cost is $W(w_1^1) = 190$. This cut will generate a constraint $x_{2,4} \leq 0$.

You can see strategy $S^4 = (0, 3, 4, 1, 4, 2, 0)$ computed in the next iteration of the algorithm in Figure 4.6. The A_1 agent has expenses $Z_1(S^4) = 350$ and agent A_2 has $Z_2(S^4) = 220$. No milestones are tardy. Two profitable increasing cuts are found in the residual networks. The first increasing cut w_1^1 is agent's A_1 and has a cost $W(w_1^1) = 120$. Agent A_1 can increase the activity on arc $(1, 2)$ and saves 120 because he pays a penalty as milestone 5 becomes tardy. The second increasing cut w_1^2 is agent's A_2 and has a cost $W(w_1^2) = 110$. Agent A_2 can increase the activity on arc $(1, 3)$ and saves 110. The program chooses cut w_1^1 and generate a lazy constraint $x_{1,2} \leq 0 + z_1$.

Next iteration of algorithm comes up with the strategy $S^5 = (0, 5, 6, 2, 4, 2, 0)$, see Figure 4.7. The A_1 agent has expenses $Z_1(S^5) = 260$ and agent A_2 has $Z_2(S^5) = 620$. Both milestones 3 and 5 are tardy. In the MILP callback function, there is 1 profitable decreasing cut w_1^1. Agent A_1 can decrease his expenses when he decreases the duration of activity $(2, 3)$ because the cost of activity $(2, 3)$ is less than a penalty for milestone 3. This cut will generate a constraint $y_{2,3} \leq 1 + z_{1,3} - z'_1$. This is an example of blocking arc $z_{1,3}$, when this arc becomes critical, the constraint has to be turned off.

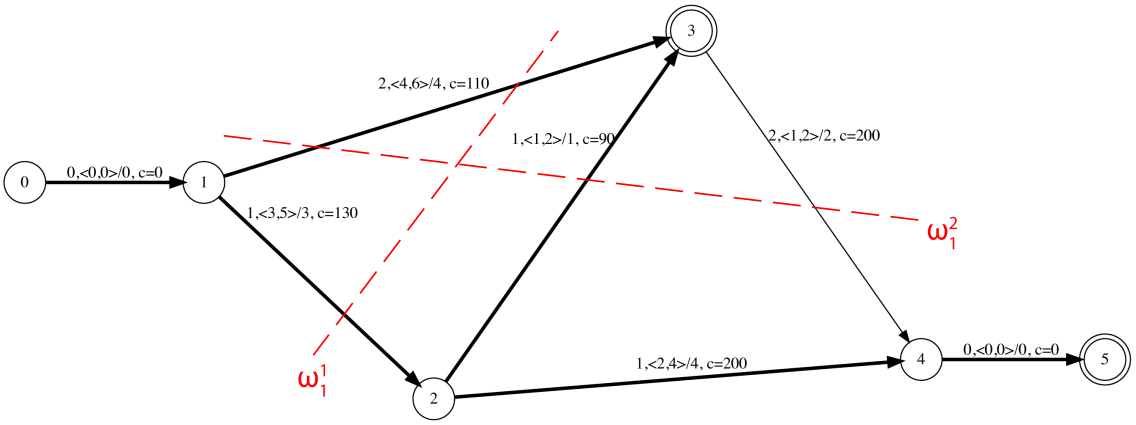


Figure 4.6: Step by step example - 5^{th} callback

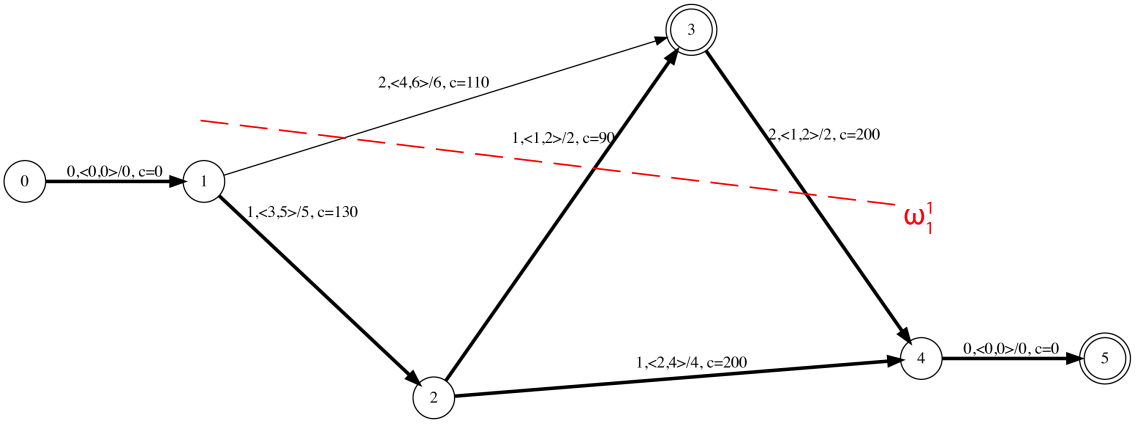


Figure 4.7: Step by step example - 6^{th} callback

Final strategy $S^6 = (0, 4, 5, 1, 4, 2, 0)$ is the optimal solution, no profitable increasing or decreasing cuts are found in the residual network for both agents. The test for the Nash solution from Section 2.1 passes correctly. We can confirm the correctness of the algorithm, in this case, using total enumeration method. From all possible combinations of processing times on arcs, we take Nash solutions and compute the makespan of the projects. The minimum value is our result.

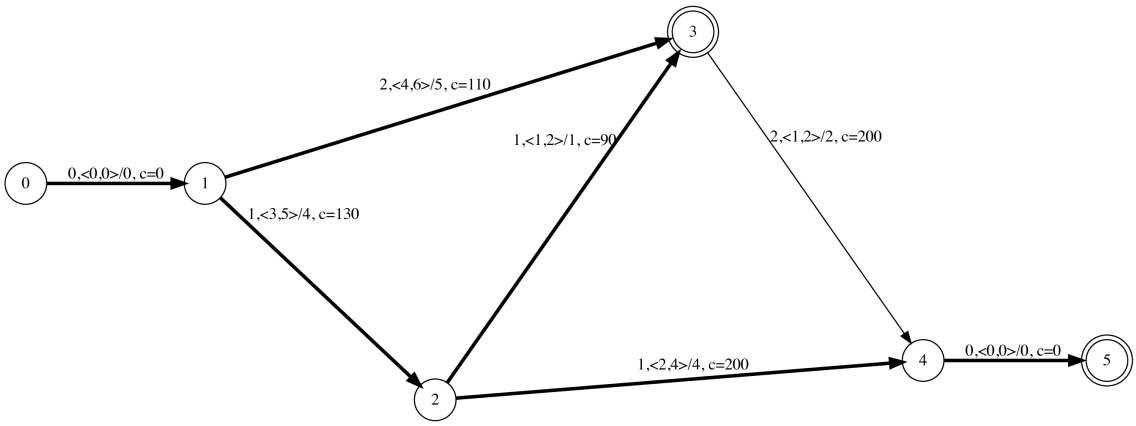


Figure 4.8: Step by step example - final solution

$\|U_R\|[-]$	avg $\|U\|[-]$	avg $\|X\|[-]$	avg $CPUtime[s]$	min $CPUtime[s]$	max $CPUtime[s]$
10	17.2100	9.9700	0.1982	0.0535	0.4620
20	50.5700	20.2800	0.8346	0.1771	2.1689
30	98.0200	33.6400	4.0767	0.9209	11.8401
40	162.4000	47.2400	20.2193	2.9263	90.1049
50	239.4800	62.2100	62.2966	7.4153	314.0791

Table 4.1: MILP model CPU run time depending on number of activities

$\|U_R\|[-]$	avg $constraints[-]$	avg $variables[-]$
10	122.2000	205.2600
20	340.0200	584.4700
30	648.0400	1 123.0200
40	1 058.1800	1 848.3000
50	1 546.7900	2 714.7800

Table 4.2: MILP model number of constraints and variables depending on number of activities

4.2 Performance

An important property of the project plan is a number of activities. In the first experiment, we measured a CPU processing time, the number of variables and constraints in MILP for $|U_R| = 10, 20, 30, 40$ and 50 activities in the project plan. Also, we measured the total number of activities (activities + dummy activities) avg$|U|$ and a total number of nodes avg$|X|$ in the problem instances. The number of agents in the instances were 3, the due date of milestones was set to 0.5, a number of milestones 0.25 and penalty of milestones 0.5. The results are summarized in Tables 4.1 and 4.2. The experiment has shown that processing time of instances is increasing with bigger size problems. We can compare the average number of variables and constraints with [9], where the problem without milestones is measured. In our model, the lazy constraint generation in MILP is used, so the average numbers of variables and constraints are significantly lower.

Figures 4.9 - 4.11 show how many instances (in %) are solved in a given amount of time for $|U_R| = 30$, 40 and 50 activities.

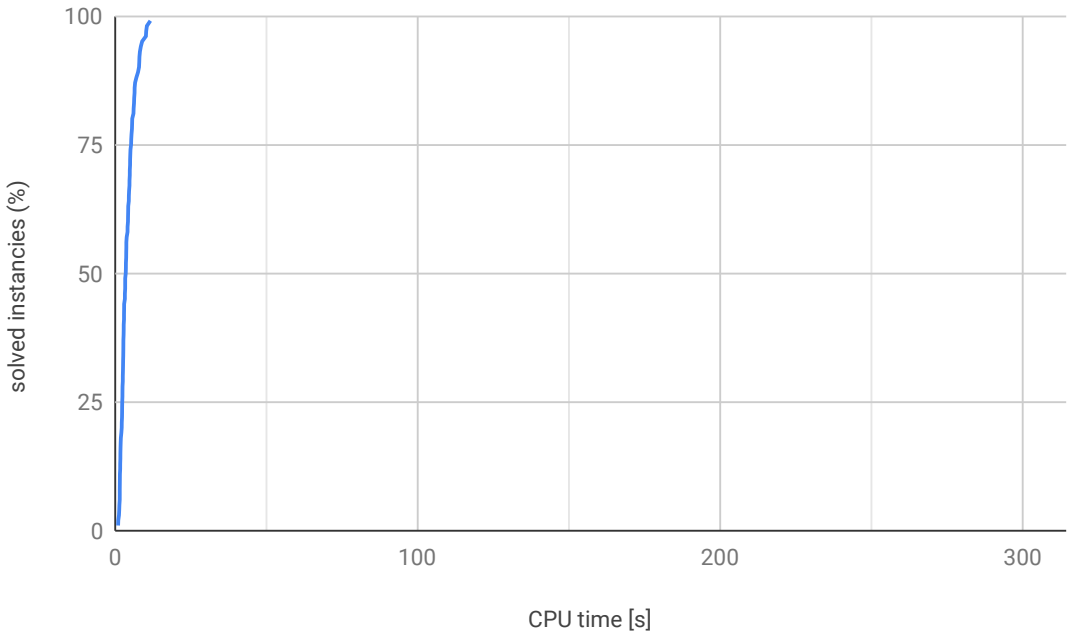


Figure 4.9: MILP model performance for $|U_R| = 30$: percentage of instances solved over time

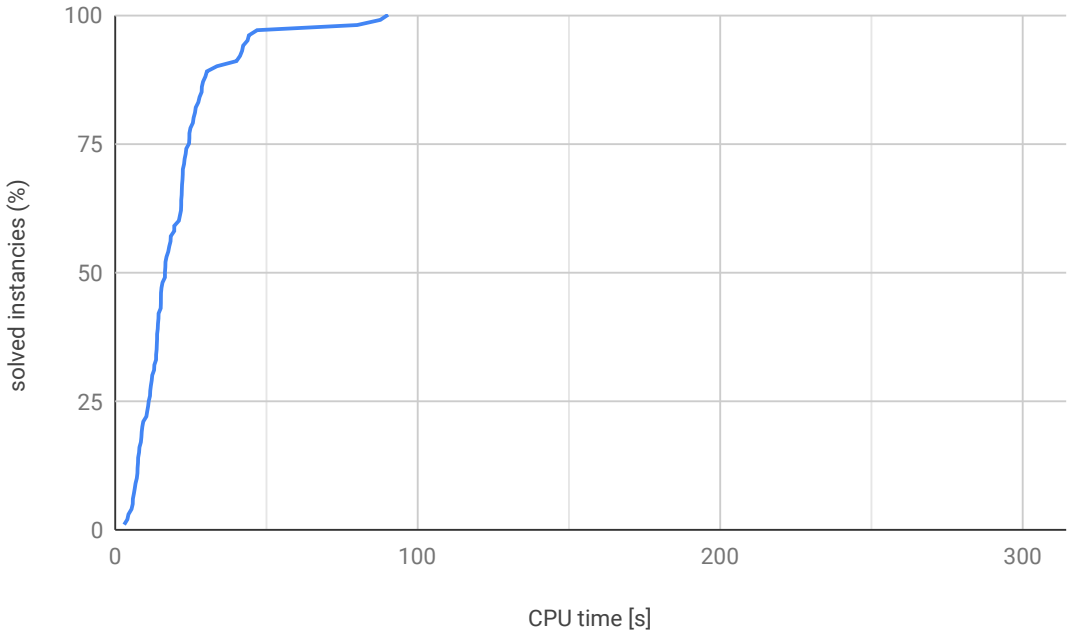


Figure 4.10: MILP model performance for $|U_R| = 40$: percentage of instances solved over time

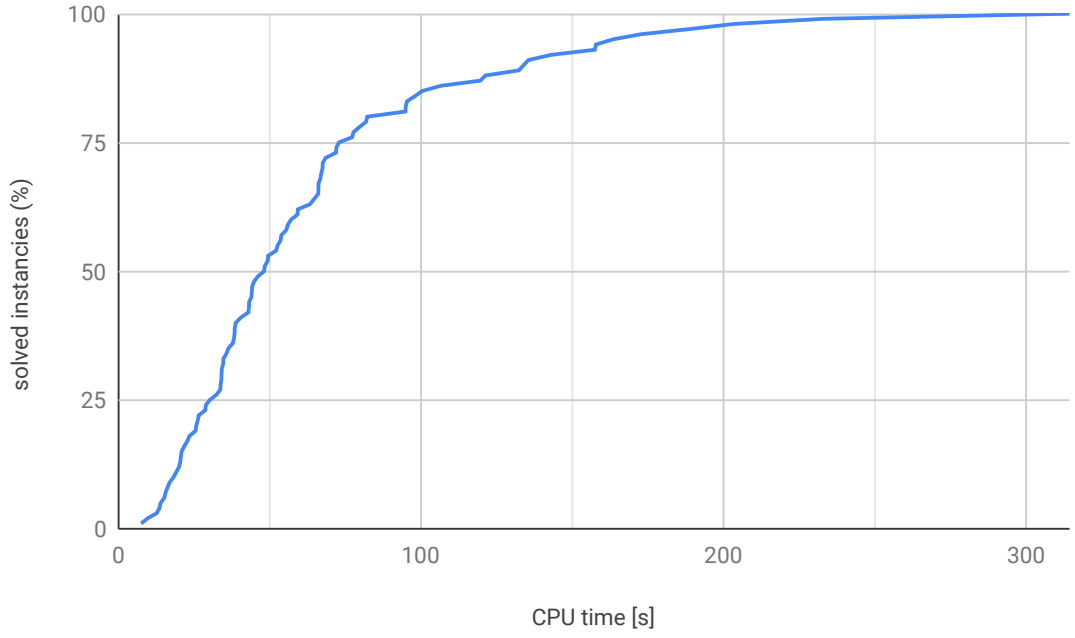


Figure 4.11: MILP model performance for $|U_R| = 50$: percentage of instances solved over time

$a[-]$	avg $CPUtime[s]$	min $CPUtime[s]$	max $CPUtime[s]$	avg $constraints[-]$	avg $variables[-]$
2	4.9040	0.6402	13.5058	648.0400	1 123.0200
4	2.7969	0.6488	9.3603	648.0400	1 123.0200
8	4.9440	0.7360	16.7203	648.0400	1 123.0200
16	9.8961	2.7834	45.3692	648.0400	1 123.0200

Table 4.3: MILP model CPU run time depending on number of agents

$N_m/N[-]$	avg $CPUtime[s]$	min $CPUtime[s]$	max $CPUtime[s]$	avg $constraints[-]$	avg $variables[-]$
0.00	0.9041	0.2146	2.4634	624.7600	1 084.2200
0.25	2.6770	0.5672	8.3990	648.0400	1 123.0200
0.50	3.7032	0.6442	13.0652	671.5300	1 162.1700
0.75	3.9914	0.6497	12.4293	694.0300	1 199.6700
1.00	3.9156	0.6710	11.8641	716.6800	1 237.4200

Table 4.4: MILP model CPU run time depending on number of milestones

Table 4.3 illustrates the influence of the number of agents on the CPU time. We measured the CPU processing time, the number of variables and constraints in MILP for instances where $a = 2, 4, 8, 16$. The due date of milestones was set to 0.5, the number of milestones 0.25 and penalty of milestones 0.5 in the problem instances. The number of agents does not affect the number of variables and constraints.

In the same way, Table 4.4 represents the influence of the number of milestones on the CPU time. We measured the CPU processing time, the number of variables and constraints in MILP for instances where the ratio of milestones to all nodes in tbe graph where $N_m/N = 0, 0.25, 0.5, 0.75, 1$. The average CPU time increased when the number of milestones increased. If we compare the time for milestones ratio 0.75 and 1, we see just a small difference, the graph is saturated with milestones.

Table 4.5 shows the influence of penalty for tardy milestones on CPU time and project makespan. The due date of milestones was set to 0.5, the number of milestones 0.25 and penalty of milestones 0.5 in the problem instances. This is a reason why, when the penalty is set to a maximal value, the ratio of project makespan is 0.5. The processing time is higher when the penalty is 0.1 and 0.2. In these cases, the algorithm will choose an activity processing time between normal duration and incompressible limit. See Figures 4.12 and 4.13 for more details.

As you see in Table 4.6, a different number of activities does not significantly affect the PoA and PoS. The reason is that the model is stabilized by milestones in the project. More agents affect the PoA, due to Table 4.7. The explanation is when more contractors are assigned to the project, all of them will try to minimize their expenses and project duration can increase. Table 4.8 shows how penalty policy affects both PoA and PoS, as the penalty for tardy milestones is getting higher, the project is getting stabilized.

$\frac{q_m}{W(\omega_{max})}[-]$	avg $CPUtime[s]$	$\frac{D(S)-\underline{D}}{\overline{D}-\underline{D}}[-]$
0.0	2.795	1.000
0.1	4.039	0.623
0.2	3.044	0.531
0.3	2.620	0.516
0.4	2.594	0.511
0.5	2.585	0.506
0.6	2.432	0.506
0.7	2.456	0.504
0.8	2.372	0.504
0.9	2.478	0.504
1.0	2.399	0.504

Table 4.5: MILP model penalty policy comparison

$\lvert U_R\rvert[-]$	$PoA[-]$	$PoS[-]$
20	1.044	1.009
30	1.048	1.001
40	1.028	1.002

Table 4.6: PoA and PoS analysis for different number of activities

$a[-]$	$PoA[-]$	$PoS[-]$
2	1.023	1.005
4	1.045	1.004
8	1.109	1.030

Table 4.7: PoA and PoS analysis for different number of agents

$\frac{q_m}{W(\omega_{max})}[-]$	$PoA[-]$	$PoS[-]$
0	1.436	1.436
0.2	1.077	1.024
0.4	1.053	1.006
0.6	1.046	1.001
0.8	1.043	1.000
1	1.038	0.999

Table 4.8: PoA and PoS penalty policy comparison

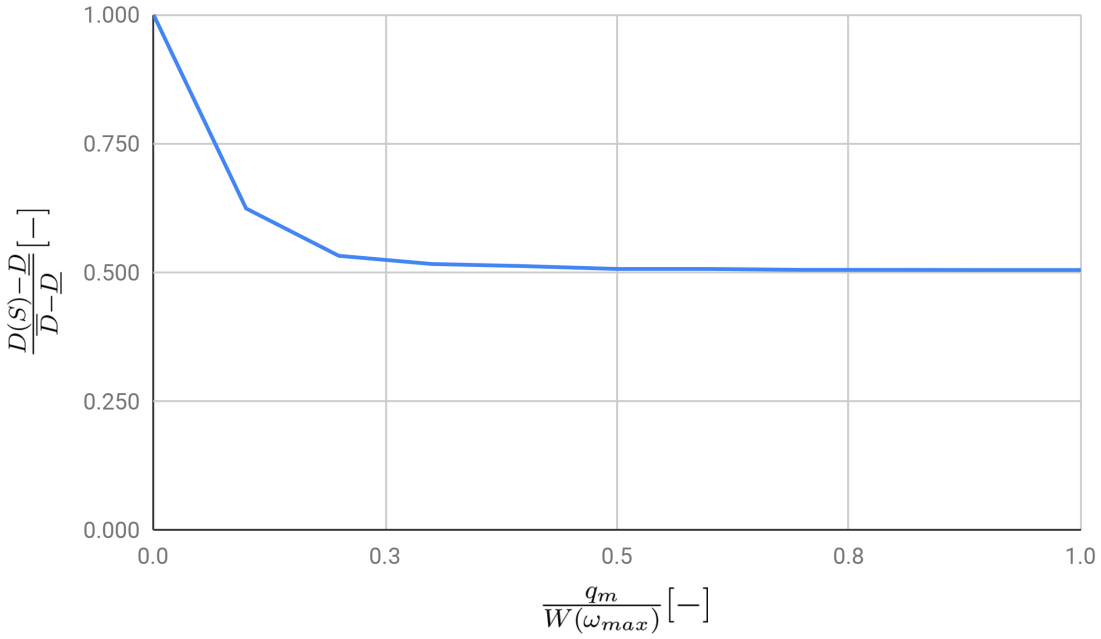


Figure 4.12: Project makespan as a function of relative milestones penalty

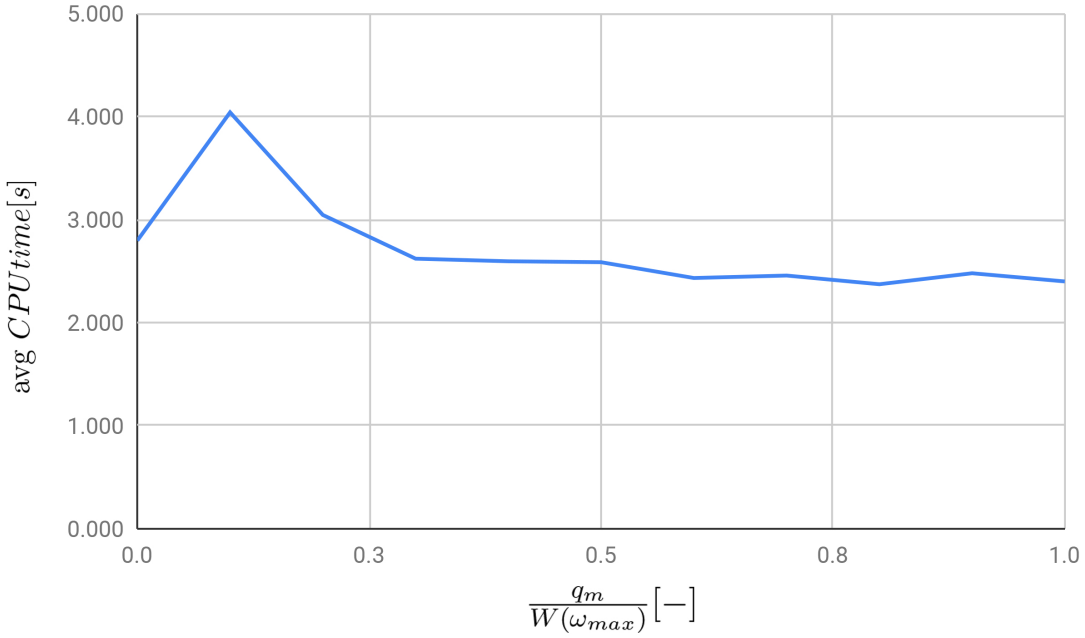


Figure 4.13: CPU tun time as a function of relative milestones penalty

Chapter 5

Conclusion

In this thesis, we extended the scope of the project scheduling problem considered in [7] by definition of milestones. The thesis provides a formal definition of an agent's behavior as MILP and a formulation of a stable solution expressed as Nash equilibria. Then it shows how a Nash equilibria can be defined via cuts in a residual graph. Also, we propose a new algorithm based on MILP and a lazy constraint generation method. A step by step program execution example is shown in the thesis. We measured the algorithm under different conditions (number of activities, agents, milestones) and proved his effectiveness. We discussed how the penalty policy for tardy milestones could affect the project duration and investigated the values of the price of anarchy and the price of stability to get useful insights for the project manager.

Future research should concentrate on the case where the amount of penalty payment is part of the decision. That may lead to a more efficient solution compare to the case with fixed penalties. Also, a generalization of the agent's objective function can be analyzed, where an agent will not be only penalized for tardy milestones, but on the other hand will get a reward for milestones reached before the due date.

Appendix A

Complete MILP formulation

A.0.1 The objective of an agent

The reaction of agent A_u to strategy profile $p(S)$:

$$\min \sum_{m \in N_m} q_{mu} T_m + \sum_{(i,j) \in \tau_u} c_{ij}(\overline{p}_{ij} - p_{ij})$$

s.t.

$$t_j - t_i - p_{ij} - s_{ij} = 0 \qquad \forall (i,j) \in E \qquad \text{(A.1)}$$

$$\underline{p}_{ij} \leq p_{ij} \leq \overline{p}_{ij} \qquad \forall (i,j) \in \tau_u \qquad \text{(A.2)}$$

$$p_{ij} = p_{ij}(S) \qquad \forall (i,j) \notin \tau_u \qquad \text{(A.3)}$$

$$t_m - d_m \leq T_m \qquad \forall m \in N_m \qquad \text{(A.4)}$$

where

$$q_{mu}, d_m, t_i, T_m, s_{ij} \in \mathbb{R}_{\geq 0}, \;\; p_{ij} \in \mathbb{Z}_{\geq 0}$$

A.0.2 Main problem

$$\min \left(t_n + \frac{\sum_{\forall (i,j)\in E} c_{ij}\left(\overline{p}_{ij} - p_{ij}\right) + \sum_{\forall m \in N_m} \sum_{\forall u \in A} q_{mu} T_m}{1 + \sum_{\forall (i,j)\in E} c_{ij}\left(\overline{p}_{ij} - \underline{p}_{ij}\right) + \sum_{\forall m \in N_m} \sum_{\forall u \in A} q_{mu}(t_m^{max} - d_m)} \right) \tag{A.5}$$

s.t.

$$t_j - t_i - p_{ij} - s_{ij} = 0 \qquad \forall (i,j) \in E \tag{A.6}$$

$$t_0 = 0 \tag{A.7}$$

$$\underline{p}_{ij} \leq p_{ij} \leq \overline{p}_{ij} \qquad \forall (i,j) \in E \tag{A.8}$$

$$\epsilon - z_{ij} \leq s_{ij} \leq \overline{s}_{ij}\,(1 - z_{ij}) \qquad \forall (i,j) \in E \tag{A.9}$$

$$z_{ij} \leq \sum_{\forall (k,i)\in E} z_{ki} \qquad \forall (i,j) \in E : i > 1 \tag{A.10}$$

$$z_{ij} \leq \sum_{\forall (j,l)\in E} z_{jl} \qquad \forall (i,j) \in E : j < n, j \notin N_m \tag{A.11}$$

$$\sum_{\forall (i,m)\in E} z_{im} \geq 1 \qquad \forall m \in N_m \tag{A.12}$$

$$1 \leq \sum_{\forall (1,i)\in E} z_{1i} \tag{A.13}$$

$$t_m \leq d_m - 1 + (t_m^{max} - t_m^{min} + 1) z_m \qquad \forall m \in N_m \tag{A.14}$$

$$t_m \geq d_m - (t_m^{max} - t_m^{min})(1 - z_m) \qquad \forall m \in N_m \tag{A.15}$$

$$t_m \leq d_m + (t_m^{max} - t_m^{min}) z'_m \qquad \forall m \in N_m \tag{A.16}$$

$$t_m \geq d_m + 1 - (t_m^{max} - t_m^{min} + 1)(1 - z'_m) \qquad \forall m \in N_m \tag{A.17}$$

$$t_m - d_m \leq T_m \qquad \forall m \in N_m \tag{A.18}$$

$$x_{ij} \leq \left(\overline{p}_{ij} - p_{ij}\right) \leq \left(\overline{p}_{ij} - \underline{p}_{ij}\right) x_{ij} \qquad \forall (i,j) \in E \tag{A.19}$$

$$y_{ij} \leq \left(p_{ij} - \underline{p}_{ij}\right) \leq \left(\overline{p}_{ij} - \underline{p}_{ij}\right) y_{ij} \qquad \forall (i,j) \in E \tag{A.20}$$

$$z_{ij} \geq x_{ij} \qquad \forall (i,j) \in E \tag{A.21}$$

$$\sum_{\forall (i,j)\in F_k} x_{ij} + \sum_{\forall (i,j)\in B_k} y_{ij} \leq |F_k \cup B_k| - 1 + \sum_{\forall (i,j)\in C_k} z_{ij} + \sum_{\forall m \in D_k} z_m \qquad \forall LC_k^{inc} \in LC^{inc} \tag{A.22}$$

$$\sum_{\forall (i,j)\in F_k} y_{ij} + \sum_{\forall (i,j)\in B_k} x_{ij} \leq |F_k \cup B_k| - 1 + \sum_{\forall (i,j)\in C_k} z_{ij} + \sum_{\forall m \in D_k} (1 - z'_m) \qquad \forall LC_k^{dec} \in LC^{dec} \tag{A.23}$$

where

$q_{mu}, d_m, t_i, T_m, s_{ij} \in \mathbb{R}_{\geq 0}$, $p_{ij} \in \mathbb{Z}_{\geq 0}$, $x_{ij}, y_{ij}, z_{ij}, z_m, z'_m \in \{0,1\}$

$LC_k^{inc/dec} = \langle F_k, B_k, C_k, D_k \rangle$, $\epsilon = 1/n$, $\overline{s}_{ij} = D(\overline{S})$.

A.0.3 Subproblems (lazy constraint generation)

Increasing Cuts

$M^C = \{m \in N_m : z_m = 1\}$

$$\max \sum_{(i,j)\in E} \alpha_{ij} l_{ij} - \sum_{(i,j)\in E} \beta_{ij} u_{ij} - \sum_{m \in M^C} q_{mu}(1-\gamma_m)$$

s.t.

$$\alpha_{ij} - \beta_{ij} = \gamma_i - \gamma_j \qquad \forall (i,j) \in E \qquad \text{(A.24)}$$

$$\alpha_{ij} + \beta_{ij} \leq 1 \qquad \forall (i,j) \in E \qquad \text{(A.25)}$$

$$\gamma_1 = 1 \qquad \text{(A.26)}$$

$$\textstyle\sum_{m \in N_m} \gamma_m \leq |N_m| - 1 \qquad \text{(A.27)}$$

where

$$q_{mu}, \ell_{ij}, u_{ij} \in \mathbb{R}_{\geq 0}, \;\; \alpha_{ij}, \beta_{ij}, \gamma_i \in \{0,1\}$$

Constraint $LC_k^{inc} = (F_k, B_k, C_k, D_k)$ where

$F_k = \{(i,j) \in E : \alpha_{ij} = 1 \wedge z_{i,j} = 1\}$,
$B_k = \{(i,j) \in E : \beta_{ij} = 1 \wedge z_{i,j} = 1\}$,
$C_k = \{(i,j) \in E : \beta_{ij} = 1 \wedge z_{i,j} = 0\}$,
$D_k = \{m \in N_m : \gamma_m = 0 \wedge z_m = 0\}$.

Decreasing Cuts

$M^C = \{m \in N_m : z'_m = 1\}$

$$\min \sum_{(i,j)\in E} \alpha_{ij} u_{ij} - \sum_{(i,j)\in E} \beta_{ij} l_{ij} - \sum_{m \in M^C} q_{mu}(1-\gamma_m)$$

s.t.

$$\alpha_{ij} - \beta_{ij} = \gamma_i - \gamma_j \qquad \forall (i,j) \in E \qquad \text{(A.28)}$$

$$\alpha_{ij} + \beta_{ij} \leq 1 \qquad \forall (i,j) \in E \qquad \text{(A.29)}$$

$$\gamma_1 = 1 \qquad \text{(A.30)}$$

$$\textstyle\sum_{m \in N_m} \gamma_m \leq |N_m| - 1 \qquad \text{(A.31)}$$

where

$$q_{mu}, \ell_{ij}, u_{ij} \in \mathbb{R}_{\geq 0}, \;\; \alpha_{ij}, \beta_{ij}, \gamma_i \in \{0,1\}$$

Constraint $LC_k^{dec} = (F_k, B_k, C_k, D_k)$ where

$F_k = \{(i,j) \in E : \alpha_{ij} = 1 \wedge z_{i,j} = 1\}$,
$B_k = \{(i,j) \in E : \beta_{ij} = 1 \wedge z_{i,j} = 1\}$,
$C_k = \{(i,j) \in E : \alpha_{ij} = 1 \wedge z_{i,j} = 0\}$,
$D_k = \{m \in N_m : \gamma_m = 0 \wedge z'_m = 1\}$.

Bibliography

[1] I. Averbakh, "Nash equilibria in competitive project scheduling", *European Journal of Operational Research*, vol. 205, no. 3, pp. 552–556, 2010.

[2] G. De Ita Luna, F. Zacarias-Flores, and L. C. Altamirano-Robles, "Finding pure nash equilibrium for the resource-constrained project scheduling problem", *Computación y Sistemas*, vol. 19, no. 1, pp. 17–27, 2015.

[3] P. Varakantham and N. Fu, "Mechanism design for strategic project scheduling", 2017.

[4] R. Van Eynde, "Multi-project scheduling—the application of a decoupled schedule generation scheme and a game mechanic", PhD thesis, Master's Dissertation in Business Engineering, Universiteit Gent, 2017.

[5] G. Confessore, S. Giordani, and S. Rismondo, "A market-based multi-agent system model for decentralized multi-project scheduling", *Annals of Operations Research*, vol. 150, no. 1, pp. 115–135, 2007.

[6] N. Chaabane, C. Briand, and M.-J. Huguet, "A multi-agent min-cost flow problem with controllable capacities: Complexity of finding a maximum-flow nash equilibrium", in *International Conference on Operations Research and Enterprise Systems (ICORES)*, 2014, 8p.

[7] A. Agnetis, C. Briand, J.-C. Billaut, and P. Šůcha, "Nash equilibria for the multi-agent project scheduling problem with controllable processing times", *Journal of Scheduling*, vol. 18, no. 1, pp. 15–27, 2015.

[8] A. Estévez-Fernández, "A game theoretical approach to sharing penalties and rewards in projects", *European Journal of Operational Research*, vol. 216, no. 3, pp. 647–657, 2012.

[9] C. Briand, S. U. Ngueveu, and P. Šůcha, "Finding an optimal nash equilibrium to the multi-agent project scheduling problem", *Journal of Scheduling*, vol. 20, no. 5, pp. 475–491, 2017.

[10] A. Agnetis, C. Briand, S. U. Ngueveu, and P. Šůcha, "Price of anarchy and price of stability in multi-agent project scheduling", *Annals of Operations Research*, pp. 1–23, 2019.

[11] S. Phillips Jr and M. I. Dessouky, "Solving the project time/cost tradeoff problem using the minimal cut concept", *Management Science*, vol. 24, no. 4, pp. 393–400, 1977.

[12] E. L. Demeulemeester and W. S. Herroelen, *Project scheduling: A research handbook*. Springer Science & Business Media, 2006, vol. 49, ISBN: 1-4020-7051-9.

[13] E. Demeulemeester, M. Vanhoucke, and W. Herroelen, "Rangen: A random network generator for activity-on-the-node networks", *Journal of scheduling*, vol. 6, no. 1, pp. 17–38, 2003.

[14] J. Ellson, E. Gansner, L. Koutsofios, S. C. North, and G. Woodhull, "Graphviz—open source graph drawing tools", in *International Symposium on Graph Drawing*, Springer, 2001, pp. 483–484.