Master Thesis

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Control Engineering

A Bi-level Solution Procedure for The Integrated Personnel Staffing and Project Scheduling Problem

Pavel Milička

Supervisor: doc. Ing. Přemysl Šůcha, Ph.D.
Field of study: Open Informatics
Subfield: Artificial Intelligence
January 2019

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení:	**Milička**	Jméno: **Pavel**	Osobní číslo: **420180**
Fakulta/ústav:	**Fakulta elektrotechnická**		
Katedra/ústav:	**Katedra počítačů**		
Studijní program:	**Otevřená informatika**		
Studijní obor:	**Umělá inteligence**		

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Dvouúrovňová optimalizace integrovaného problému rozvrhování liských zdrojů a projektu

Název diplomové práce anglicky:

A Bi-level solution procedure for the integrated personnel staffing and project scheduling problem

Pokyny pro vypracování:

1) Propose a proper decomposed problem formulation to integrate the personnel staffing problem and the project scheduling problem by making use of its problem structure.
2) Devise a decomposition optimization algorithm to optimize the quality of the personnel roster and project schedule simultaneously in order to overcome the shortcoming of an isolated composition of schedules. This implies that both the activity start times and the best mix of resources (amount and type) in terms of cost should be determined given a particular project deadline.
3) Adopt benchmark instances described in [2] and use them for the algorithm benchmarking. If possible, benchmark the algorithm with state-of-the-art results.

Seznam doporučené literatury:

[1] Maenhout, B. and Vanhoucke, M., 2016, "An exact algorithm for an integrated project staffing problem with a homogeneous workforce?, Journal of Scheduling, 19, 107-133 (doi:10.1007/s10951-015-0443-z).
[2] Maenhout, B. and Vanhoucke, M., 2015, "A Resource Type Analysis of the Integrated Project Scheduling and Personnel Staffing Problem?, Annals of Operations Research, To Appear, (doi: 10.1007/s10479-015-2033-z).
[3] Briand, C. - Ngueveu, S.U. - Šůcha, P. 2017, "Finding an optimal Nash equilibrium to the multi-agent project scheduling problém?, Journal of Scheduling. 20(5), 475-491 (doi: 10.1007/s10951-017-0516-2).

Jméno a pracoviště vedoucí(ho) diplomové práce:

doc. Ing. Přemysl Šůcha, Ph.D., optimalizace CIIRC

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

Datum zadání diplomové práce: **01.08.2018** Termín odevzdání diplomové práce: ____________

Platnost zadání diplomové práce: **30.09.2019**

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

3/1/2019
Datum převzetí zadání

Podpis studenta

Acknowledgements

I would like to thank both of my supervisors, doc. Ing. Přemysl Šůcha, Ph.D. from the Czech Technical University in Prague and prof. Broos Maenhout from the Ghent University, for their guidance, advice and patience.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, 8 January 2019

....

Pavel Milička

Abstract

This thesis aims to propose a model integrating a project scheduling and personnel staffing problem along with an algorithm solving it. A bilevel optimization framework is utilized to formulate the problem, modelling interaction between different entities each pursuing individual objectives.

A novel algorithm solving the bilevel model based on lazy constraints generation is proposed. The algorithm employs two types of lazy constraints, where both are used to restrict the solution space to comply with the concept of bilevel feasibility. However, only the first type is necessary, the second type is used as a speed-up technique utilizing the previously discovered solutions.

The proposed problem is compared with the single-level approaches solving the same combination of objectives to show the uniqueness of the bilevel approach. Experimental evaluation is conducted to show the influence of the instance and algorithm properties on the algorithm's performance and solution quality.

Keywords: project scheduling, RCPSP, bilevel programming, optimization, lazy constraints

Supervisor: doc. Ing. Přemysl Šůcha, Ph.D.

Abstrakt

Cíl této práce je navrhout model integrující rozvrhování projektů a najímání zaměstanců spolu s algoritmem, který by tento problém řešil. Pro formulaci problému je využito dvojúrovňové programování, které modeluje interakce mezi různými entitami, kdy se každá snaží dosáhnout individuálního cíle.

Je navrhnut algoritmus založený na tvorbě tzv. "líných" omezujících podmínek řešící tento dvojúrovňový model. Algoritmus využívá dva typy těchto omezujících podmínek, kde oba jsou použity k omezení prostoru řešení tak, aby splňoval koncept dvojúrovňové přípustnosti. Nicméně jen první typ těchto omezujících podmínek je nutný, druhý je použit pro zrychlení algoritmu a využívá předchozí řešení.

Navržený problém je porovnán s jednoúrovňovými přístupy řešící stejnou kombinaci kritérií, aby se ukázala unikátnost dvojúrovňového přístupu. Je provedena experimentální evaluace pro ukázání vlivu vlastností instancí a algoritmu na výkonu a kvalitě řešení.

Klíčová slova: rozvrhování projektů, RCPSP, dvojúrovňová optimalizace, optimalizace, lazy constraints

Contents

1 Introduction **1**

1.1 Related work 2

1.1.1 Project scheduling 2

1.1.2 Bilevel programming 3

1.2 Thesis outline 4

2 Theoretical Background **5**

2.1 Resource constrained project scheduling 5

2.1.1 Project components 5

2.1.2 Objectives 7

2.1.3 Graph representation 8

2.1.4 Linear programming representation 9

2.2 Bilevel optimization 10

2.2.1 Bilevel formalism 10

2.2.2 Classification of the BLPP . . 12

3 Problem Formulation **15**

4 Proposed Algorithm **19**

4.1 Initialization 19

4.2 High point problem 21

4.3 Bilevel feasibility callback 21

4.4 Algorithm summary 23

5 Comparison with Single-level Approaches **25**

5.1 Sequential approach 25

5.2 Multi-objective approach 28

5.3 Comparison of all approaches . . 31

5.4 Summary 33

6 Experiments **35**

6.1 Datasets 35

6.1.1 RANGEN2 36

6.1.2 Patterson dataset 36

6.2 External validation 37

6.3 Internal validation – speed-ups . 39

6.4 Internal validation – parameter setting 43

6.5 Internal validation – problem instances 45

6.6 Summary 47

7 Conclusion and Future Work **49**

7.1 Future work 49

Bibliography **51**

A CD Contents **57**

B Abbreviations **59**

Figures

2.1 Example of an Activity on Node project representation 9

2.2 Inducible regions of different linear bilevel problem categories 14

4.1 Proposed algorithm flow chart . . 20

5.1 Schemas of the different sequential algorithm 26

5.2 The problem instance used for proving the difference between the bilevel and integrated approach. . . 29

5.3 Schemas of the different sequential algorithm 30

5.4 Leader's objective value of the different approaches for each makespan of the counter-example. 33

6.1 Patterson dataset properties . . . 37

6.2 Runtime comparison of different approaches 38

6.3 Objective value comparison of different approaches 38

6.4 Solution components comparison of different approaches 39

6.5 Influence of LC2 on the performance 41

6.6 Comparison of different branching strategies 42

6.7 Influence of using the EST and LST on the performance 42

6.8 The algorithm's performance under various combinations of α and β . . 43

6.9 The algorithm's performance with different values of R 44

6.10 The algorithm's performance with different number of activities 45

6.11 The algorithm's performance with different values of the series-parallel indicator 46

6.12 The algorithm's performance with different number of resource types 46

Tables

5.1 Comparison of the sequential and bilevel approach 27

5.2 Iterations of the integrated approach trying to find the same solution as the bilevel one 30

5.3 Iteration of the integrated approach trying to achieve a makespan of $\delta = 55$ 32

6.1 Comparison of different approaches 40

Chapter 1

Introduction

Project scheduling is a widespread problem occurring in various areas from building industry to software development. The cornerstone of a successful project management from the business point of view is balancing the time, quality and cost triangle which is directly connected to human resources management [13]. The quality of a project schedule can be evaluated by different performance measures [11], e.g., by completion time, additional employees hired, levelling of the resource demand, financial objectives and many others. The common approach to approximate the decision making process is to formulate the problem either as a single-level optimization with single or multiple objective functions or to employ some kind of sequential scheme solving one objective after another to come to a final schedule.

However, both single-level and sequential formulations fail to model a real-world scenario, where multiple entities pursue different objectives and need to come to an agreement on the resulting project schedule, as the former case models the problem with a single entity having all the knowledge and optimizing the linear combination of goals, and the latter models a scenario with different entities, where the entity solved previously overrules the other ones remaining in the sequence.

The thesis tries to model the interaction of different entities, which need to come to an agreement on the result, by formulating the problem by means of bilevel programming framework while showing that using the other approaches can lead to different solutions. The detailed description of the considered scenario is given below.

Every project requires a team for executing the project's activities, its management and decision making regarding the project schedule [11]. The team may have various structures to suit the specifics of the company and/or the project. The thesis considers a project-based team structure [7] as it provides solid background for formulating the problem as a bi-level optimization problem while following a real-world scenario. The considered team structure is hierarchical and consists of three position types: a *team leader*, a *project* *manager* and *employees* of various skills. The *team leader* is standing at the top of the team hierarchy and determines whether to hire additional employees of certain skill while trying to keep the workload of employees (resource requirements) as leveled as possible. At the lower layer of the hierarchy is the *project manager*, who addresses the project scheduling, i.e., who is responsible for determining the start and end times of project's activities and who tries to minimize the project's makespan. The project's activities are then executed by the lowest level of the hierarchy, the *employees* already present in the team or hired by the team leader.

The aim of this thesis is to

- formulate the problem integrating both personnel staffing and project scheduling by making use of the problem's hierarchical structure
- propose an algorithm providing both project schedule and employees composition
- benchmark the algorithm with state-of-the-art results

The thesis contributes to the existing literature on project scheduling and personnel staffing by modelling the problem via bilevel optimization, thus, presenting a novel point of view on the problem, where exists a competition between the different entities involved.

Also, the thesis contributes significantly to the existing algorithms solving integer bilevel problems as to the best of our knowledge, a lazy constraint approach to solving bilevel problems has not yet been used.

1.1 Related work

The relevant literature can be divided into two parts. The first part considers the problem from the view of project scheduling and gives an overview of a relevant literature to be found on the topic of resource-constrained project scheduling (RCPSP) and integrated project scheduling and personnel staffing. The second part gives relevant literature regarding bilevel optimization techniques while focusing on algorithms applicable on discrete bilevel problems.

1.1.1 Project scheduling

From the point of view of project scheduling, dealing with workforce without taking into account days-on/days-off schedules of employees occurs in resource-constrained project scheduling [8], [23] where the project needs resources for its activities execution and employees can be considered as one of the possible resources.

Surveys of resource-constrained project scheduling methods can be found in [34, 24, 26]. The most common type of the RCPSP [11] tries to minimize the overall project makespan while satisfying the constraints given by resources availability. In [22], the possible extensions of the RCPSP as multi-mode activities, minimal lags, time-varying resource demands etc. are surveyed. In [45], the authors proposed a method solving the RCPSP with logical constraints employing SAT solvers. Heuristics solving the RCPSP and their experimental evaluation are presented in [25]. Use of genetic algorithms to solve various project scheduling problems are proposed in [42, 32, 21].
Other problems belonging in the RCPSP framework are given below. The resource availability cost problem (RACP) considers resources having associated costs and tries to minimize the resource cost while satisfying a pre-given project deadline [43]. Multi-skilled workforce is considered in [4], where activities require that a person with certain skill level is available during the activity duration. The RCPSP with discounted cashflow (RCPSPDC) assumes cash inflows and outflows during the project, where inflows occur usually when some milestone is achieved and outflows are given by using the resources and workforce. The objective of these problems is to optimize the net present value of the project and its description and algorithm for solving both single and multi-mode RCPSPDC problems is presented in [28].

Some papers try to solve the integrated project scheduling and personnel staffing problem in more detailed manner providing both project schedule and employee roster while satisfying conditions regarding maximal length of consecutive working days, minimal number of working days per time period and introducing possibility to assign overtime to employees. Example of this approach can be found in [29], where the authors used a branch and price algorithm [3] to solve the combined problem of project and personnel scheduling. In [30], the authors examine the influence of various factors in the integrated scenario, comparing cyclic and non-cyclic resources and measuring the impact of using overtime and temporary workforce on the schedule cost.

1.1.2 Bilevel programming

The idea of modelling with using two hierarchical optimization levels first appeared in [50] and comes from the notion of a Stackelberg game [49]. The Stackelberg game assumes two types of players – a leader and followers. In a Stackelberg game, the leader moves first and the followers react on its decisions. The leader has information about the followers' objectives, therefore, it can anticipate their responses and adjust its strategy accordingly. Therefore, the leader's optimization problem is a nested one, where the lower level optimization problems represent the best responses of the followers to the leader's decision. From the perspective of mathematical modelling, bilevel approaches started to draw attention with the rise of computers in nineteen-eighties with first survey being [27].

Bilevel programming is used for modelling problems that employ the hierarchical structure appearing in the Stackelberg game or when a single-level optimization cannot be utilized, e.g. in toll setting problem given in [10], where the leader representing the road management needs to determine which roads should be tolled and the follower simulates the behaviour of the drivers trying to minimize the travel costs including tolls along with other factors as duration, length of the journey, etc. Methods for solving continuous bilevel problems are given in [10]. In [41], the author also surveys algorithms that can be used for bilevel problems involving integrality of some variables along with utilization of genetic algorithms for solving bilevel problems. More methods and references for solving the bilevel problems are presented in the next chapter as the algorithms usually relate to the specific type of the bilevel problem determined by integrality of variables, which is described in more detail there.

Literature considering project scheduling in the bilevel framework is sparse. The problem of interaction between the project owner and the contractor in a fuzzy environment is considered in [51], which is solved by a particle swarm optimization technique. Multi-mode RCPSP with uncertainty is formalized as a bilevel multi-objective problem and solved by another swarm optimization algorithm in [17].

1.2 Thesis outline

The thesis is organized as follows. Chapter 2 gives a theoretical background on core concepts of the RCPSP and bilevel programming, which is crucial for understanding the problem formulation given in Chapter 3. Algorithm proposed for solving the problem is presented in Chapter 4. Proofs that the bilevel approach differs from the single-level approaches are given in Chapter 5. Experimental evaluation of the algorithm is given in Chapter 6. Finally, concluding remarks and comments about future work are dedicated to Chapter 7.

Chapter 2

Theoretical Background

Aim of this chapter is to provide the necessary background for understanding the problem formulation and the algorithm proposed to solve the problem. The chapter is divided into two sections, one describing the basics of resource constrained project scheduling problems (RCPSP) and the other the core concepts of bilevel optimization.

2.1 Resource constrained project scheduling

Definition of a project by the ISO standard [16] is following:

> Project is an unique process consisting of a set of coordinated and controlled activities with start and finish dates, undertaken to achieve an objective conforming to specific requirements, including the constraints of time, cost and resources.

Following subsections give a brief overview of the essential parts of a project as given in the definition and the options that could be considered for modelling it.

2.1.1 Project components

As mentioned in the project definition, a project consists of a set of coordinated *activities*. Activities have to be executed according to a set of various *precedence relation* constraints and require certain *resources* for their execution. Non-preemption of the project activities is usually considered, meaning once an activity of a project starts, it cannot be interrupted until it is completed. The description of a process that breaks down a project into a set of activities is beyond the scope of this thesis and can be found in [11].

Precedence relations

Precedence relation defines a relation between two activities specifying constraints for the project schedule. Precedence relations types are: start-start (SS), start-finish (SF), finish-start (FS) and finish-finish (FF). All of the above mentioned precedence relations belong to so called generalised precedence relations [14] and represent either minimal or maximal time lag between two activities i and j in a project.

Minimal time lag relations ($SS_{ij}^{\min}(x)$, $SF_{ij}^{\min}(x)$, $FS_{ij}^{\min}(x)$, $FF_{ij}^{\min}(x)$) specify the minimal time x that needs to pass between the start/end of activity i and the start/end of activity j. Maximal time lag relations ($SS_{ij}^{\max}(x)$, $SF_{ij}^{\max}(x)$, $FS_{ij}^{\max}(x)$, $FF_{ij}^{\max}(x)$) specify the maximal time x that is allowed to pass between the start/end of activity i and the start/end of activity j.

It is easy to come up with real-world scenarios for each of the relations described above and on top of that, some of the relations can be combined together, e.g., $SS_{ij}^{max}(x)$ and $FF_{ij}^{max}(x)$ specifying that the time-lag between activities i and j should never exceed x time units.

Within this notation, the simplest and the most commonly used relation type is denoted as $FS_{ij}^{min}(0)$ meaning that activity j can start as soon as all its predecessors are finished. This relation type is the only one considered in the thesis, i.e., this is the type meant by precedence relation in the rest of the text.

Resources

Activities in a project may require resources for their execution. Many of different resources can be considered, e.g., manpower, machine availability, money, space, energy consumption, etc. In terms of renewability of the resource, three resource categories can be distinguished.

First, the *renewable resources* are available by the full amount at each time step. In order for the project schedule to be feasible, the overall resource requirement of activities must not to exceed the resource availability at any time step. The best example of renewable resource is personnel, where multiple disjoint employee types exist and each activity requires a certain number of employees with the given skills working on it. Renewable resource type is denoted as $k \in K$, where K is the set of all the renewable resource types used in the problem. Constant R_k determines how many resources of type k are available at each time step and finally, r_{ik} is the amount of renewable resource of type k activity i requires at each time step of its execution.

Second, *non-renewable resource* availability is given for the whole project and the activities have to be scheduled in such a way the project fits within the resource limit. Money is an example of non-renewable resource, where a project has to fit into specified budget.

Third, *doubly-constrained resources* are a combination of the previous two types. This resource category is constrained by both time step availability of the renewable resource and overall limitation of the non-renewable one. Example of this can be available workforce along with maximum number of man-days spent on a project.

Activity duration

Each activity has a specified duration or an estimation of the duration. There are different possibilities of modelling the activity duration, where some considers the duration fixed and others take probability and chance into account drawing the duration from a probability distribution.

Deterministic activity duration assumes that each activity of a project can be estimated with absolute certainty. Activities can either be executed in a single-mode scenario, where the duration and the resource usage is given for each activity i by constant d_i, or in a multi-mode scenario, where the activity duration is a function of resources used. This allows to investigate resource-time trade-offs determining how expensive would be to decrease the project makespan. Example of solving the multi-mode resource constrained problem is given in [1].

Stochastic activity estimation assumes that the duration cannot be set with absolute confidence and that uncertainty needs to be included in the models. The PERT (Project Evaluation and Review Technique) model [31] propose the use of three estimates for each activity duration – the optimistic, pessimistic, and most likely approach. Another option is to model the durations with simple probability density functions – uniform and triangular – as proposed in [15]. Last but not least, modelling with fuzzy sets and fuzzy numbers is possible in order to account for the imprecision in the activity length estimation. The examples of using fuzzy approach in project scheduling can be found in [2, 20].

2.1.2 Objectives

Aside from constructing a feasible schedule, which is a schedule complying with both precedence and resource constraints, there is usually additional objective representing the important features of the schedule.

The most common objective is to minimize project completion time as it reflects the real-world scenario where reducing the project makespan minimizes the risk of violating a deadline. Other time-related objectives were suggested as well, e.g., minimizing lateness, which is a difference between the task completion time and its due date, minimizing tardiness, earliness, or some alterations of these measures, e.g., minimizing maximal tardiness.

Apart from the previous performance measures which are also called time-based objectives, there also exist objectives based on resources. A resource availability cost problem (RACP) solves the minimal amount of renewable resource hired in order to comply with a pre-specified project deadline. Another example is a resource levelling problem (RLP), where the level of resource requirement throughout the considered time should be as flat as possible without violating the project deadline. The levelling problem can be modelled in different ways, e.g., by minimizing the total number of resource jumps, by minimizing the squared deviations of the resource requirements from the average, etc.

Last major type of project scheduling performance measures are financial objectives. Cash inflows and outflows are considered in maximization of net present value, where completion of certain activities result in cash inflow and using resources and executing the project activities results in cash outflow. Example of this approach can be found in [38, 47].

All the project performance measures can be divided into two categories – regular and non-regular. A regular measure of performance is a non-decreasing function of the activity completion times, and therefore, these measures imply the shortest project makespan possible. On the other hand, non-regular measures of performance, also referred to as free completion measures, have no such implication, thus, making the project longer may decrease the project objective value.

More detailed description about the performance measure and their examples can be found in [23, 11].

2.1.3 Graph representation

Projects can be represented as a graph $G = (N, A)$, where N is the set of activities and A is the set of edges containing the precedence relations of type $FS_{ij}^{min}(0)$, i.e., if there exists an edge from node $i \in N$ to $j \in N$, then j can start only after the activity i is finished. This representation is called *Activity on Node* representation and allows to visualize the project in a human-readable form and work with the project within the context of a well defined data structure. The Activity on Node representation uses two additional nodes, which are called dummy nodes and are used to model the project start and the project end both having zero-length duration. This way, it is possible to model parallel activities in the start or in the end of the project. The thesis uses the Activity on Node representation exclusively. An example of a project instance graphical representation can be found in Figure 2.1.

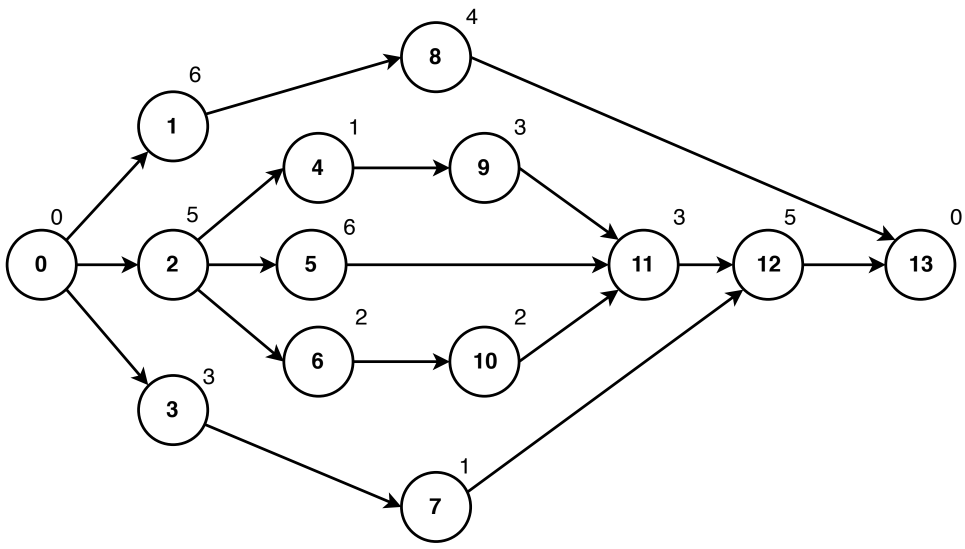


Figure 2.1: Example of an Activity on Node project representation. The nodes represent the activities with activity 0 and 13 being the dummy activities. The number at the top of the node denotes the activity duration. The project instance is taken from [44].

The other possibility of representing a project via graph is to use *Activity on Arc* format, switching the context of nodes and edges. The details about constructing the Activity on Arc representation can be found in [11].

2.1.4 Linear programming representation

The thesis follows the formulation of the RCPSP originally presented in [37], however, instead of using variables to denote the fact activity has finished, variables that are used – v_{it} – denote that the activity started at a given time.

$$v_{it} = \begin{cases} 1 & \text{if activity } i \text{ starts at time } t \\ 0 & \text{else} \end{cases} \tag{2.1}$$

The common parts of the RCPSP are following:

$$\min f(\mathbf{x}) \tag{2.2}$$

$$\text{s.t.} \sum_{t \in T} v_{it} = 1 \qquad \forall i \in N \tag{2.3}$$

$$\sum_{t \in T} t v_{jt} - \sum_{t \in T} t v_{it} \geq d_i \qquad \forall (i,j) \in A \tag{2.4}$$

$$\sum_{i \in N} \sum_{t'=t-d_i}^{t} r_{ik} v_{it'} \leq R_k \qquad \forall k \in K, \forall t \in T \tag{2.5}$$

$$v_{it} \in \{0,1\} \qquad \forall i \in N, \forall t \in T \tag{2.6}$$

Function $f(\mathbf{x})$ denotes the objective function, which differs according to the selected objective for the schedule as discussed previously. Vector of all variables of the problem is denoted as $\mathbf{x}$ and consists of $\mathbf{v}$ and other variables needed for the specific problem and performance measure formulation. Equation (2.3) forces all the activities to be executed in the schedule. The precedence constraints are enforced in Equation (2.4). The compliance with the renewable resource requirements is given in Equation (2.5).

Note that the specific form of the constraints and additional variables used in the model fully depends on the performance measure and the problem that is being modelled. As an example, if a RACP problem were to be solved, the model would contain variables denoting the number of resources bought and would also require changes in the equations formulating the resource requirements compliance. To solve the resource levelling problem, the model would require additional variables to represents the resource "levelness" both in the performance measure function and the constraints.

2.2 Bilevel optimization

The bilevel programming is essentially an optimization problem with another optimization problem nested in the constraints of the first one. The outer level is also called upper level and the inner level is called lower level. Each level has its own set of variables it controls. The two levels of the model are hierarchically organized with each level representing one entity (player) – the upper level player is called a leader and it communicates the variables it controls to the lower level entities – the followers. The leader's objective function can contain variables from both levels, whereas the follower optimizes only among its own variables with the leader's variables influencing its constraints.

2.2.1 Bilevel formalism

The thesis restricts the bilevel programming problems considered only to the linear cases, i.e., all objective functions and constraints are linear. Details about optimizing non-linear bilevel problems can be found in [9, 19, 40]. The formulation uses the following notation to formalize the bilevel linear programming problem (BLPP):

- $\mathbf{x}$, $\mathbf{y}$ – decision vectors of the upper and lower level, respectively.
- $F(\mathbf{x}, \mathbf{y})$, $f(\mathbf{y})$ – objective functions of the upper and lower level, respectively.
- $\mathbf{c}^{\mathbf{U}}_{\mathbf{x}}$, $\mathbf{c}^{\mathbf{U}}_{\mathbf{y}}$ – cost vectors for the upper level objective function associated with upper and lower level variables, respectively.

- $\mathbf{c_y^L}$ – cost vector of the lower level objective function.
- $g_x(\mathbf{x})$, $g_y(\mathbf{y})$, $g_{x,y}(\mathbf{x}, \mathbf{y})$ – constraints given for the upper, lower and both levels of the optimization problem, respectively, with associated right-sides vectors $\mathbf{b_x}$, $\mathbf{b_y}$, $\mathbf{b_{xy}}$.

With the notation being introduced, it is possible to formulate the BLPP with Equations (2.7)-(2.12). Note that the vector multiplication used in the formulation is a dot product of the vectors involved.

$$\min F(\mathbf{x}, \mathbf{y}) = \min \mathbf{c_x^U x} + \mathbf{c_y^U y} \tag{2.7}$$
$$\text{s.t. } \mathbf{x} \in X = \{\mathbf{x} : g_x(\mathbf{x}) \geq \mathbf{b_x}\} \tag{2.8}$$
$$\min f(\mathbf{y}) = \min \mathbf{c_y^L y} \tag{2.9}$$
$$\text{s.t. } g_{xy}(\mathbf{x}, \mathbf{y}) \geq \mathbf{b_{xy}} \tag{2.10}$$
$$\mathbf{y} \in Y = \{\mathbf{y} : g_y(\mathbf{y}) \geq \mathbf{b_y}\} \tag{2.11}$$
$$\mathbf{x} \geq 0, \mathbf{y} \geq 0 \tag{2.12}$$

In order to define the optimal solution of the bilevel linear programming problem, following regions and sets are defined based on [33]:

- Bilevel linear programming problem constraint region Ω is a set of all pairs $(\mathbf{x}, \mathbf{y})$ satisfying all the constraints.

$$\Omega = \{(\mathbf{x}, \mathbf{y}) : \mathbf{x} \in X, \mathbf{y} \in Y, g_{xy}(\mathbf{x}, \mathbf{y}) \geq \mathbf{b_{xy}}\} \tag{2.13}$$

- Projection of Ω to leader variable space $\Omega(X)$ is a subset of the leader's search space such that there exists a pair $(\mathbf{x}, \mathbf{y})$ in Ω for each vector $\mathbf{x}$ inside the set.

$$\Omega(X) = \{\mathbf{x} \in X : \exists \mathbf{y} \text{ such that } (\mathbf{x}, \mathbf{y}) \in \Omega\} \tag{2.14}$$

- Follower's feasible region $\Omega(\overline{\mathbf{x}})$ for specific leader's decision vector $\overline{\mathbf{x}} \in X$ is a subset of possible responses of the follower to the given leader's decision vector.

$$\Omega(\overline{\mathbf{x}}) = \{\mathbf{y} \in Y : g_{xy}(\overline{\mathbf{x}}, \mathbf{y}) \geq \mathbf{b_{xy}}\} \tag{2.15}$$

- Follower's best response set $M(\overline{\mathbf{x}})$ for a given leader's decision vector $\overline{\mathbf{x}}$ contains all the follower's variable vectors in the follower's feasible region for that vector that are optimal in terms of its objective function.

$$M(\overline{\mathbf{x}}) = \{\mathbf{y} \in Y : \text{argmin}_{\mathbf{y'}} f_y(\mathbf{y'}) \text{ where } \mathbf{y'} \in \Omega(\overline{\mathbf{x}})\} \tag{2.16}$$

- Inducible region IR consists of all pairs $(\mathbf{x}, \mathbf{y})$ such that there exists a solution vector $\mathbf{y}$ for each $\mathbf{x}$ and $\mathbf{y}$ belongs to the follower's best response set.
$$IR = \{(\mathbf{x}, \mathbf{y}) : \mathbf{x} \in \Omega(X), \mathbf{y} \in M(\mathbf{x})\} \tag{2.17}$$

Pairs $(\mathbf{x}, \mathbf{y})$ that belong to the inducible region are also called *bilevel feasible*. With the introduced notation it is finally possible to define the optimal solution of the bilevel linear programming problem:

Pair $(\mathbf{x}^*, \mathbf{y}^*)$ is called *bilevel optimal solution* if it is bilevel feasible and $F(\mathbf{x}^*, \mathbf{y}^*) \leq F(\mathbf{x}, \mathbf{y})$ for all $(\mathbf{x}, \mathbf{y}) \in IR$.

2.2.2 Classification of the BLPP

Optimistic vs pessimistic position

Consideration of two optimization levels can bring additional ambiguity to the problem, namely when the follower has multiple optimal solutions of the lower level optimization objective. In such cases two approaches are recognized, the *optimistic* and *pessimistic position*.

Most articles assume the optimistic position as it is more tractable compared to the pessimistic position [41]. The optimistic position assumes some cooperation of the players. The leader expects that the follower, who has multiple choices of $\mathbf{y} \in M(\mathbf{x})$ for the decision vector $\mathbf{x}$ selected by the leader, chooses one that benefits the leader the most.

In case of the pessimistic position, leader optimizes for the worst case that could happen, i.e., it assumes that the follower selects such decision vector from its best response set, which is least beneficial for the leader's problem. Pessimistic positions are harder to solve, as some of the techniques from dealing with optimistic positions cannot be used (e.g. single level reduction of lower-level problems) and generally the pessimistic formulations have stricter assumptions regarding the existence of the optimal solution [41].

Integrality

Another possible categorization of the BLPP raises from imposing integrality on the variables. From this perspective, four categories can be distinguished (not including the mixed-cases):

- *continuous-continuous* – the problem contains only continuous variables, i.e., $\mathbf{x} \in \mathbb{R}, \mathbf{y} \in \mathbb{R}$

- *continuous-discrete* – the leader's variables are continuous and the follower's variables are discrete: $\mathbf{x} \in \mathbb{R}, \mathbf{y} \in \mathbb{Z}$
- *discrete-continuous* – the leader's variables are discrete and the follower's variables continuous: $\mathbf{x} \in \mathbb{Z}, \mathbf{y} \in \mathbb{R}$
- *discrete-discrete* – both levels are discrete: $\mathbf{x} \in \mathbb{Z}, \mathbf{y} \in \mathbb{Z}$

An example used in [48] is used to illustrate difficulties introduced by integer variables. The example optimizes the following lower level objective

$$\min_{y} y \tag{2.18}$$

$$\text{s.t. } x + y \leq 2 \tag{2.19}$$

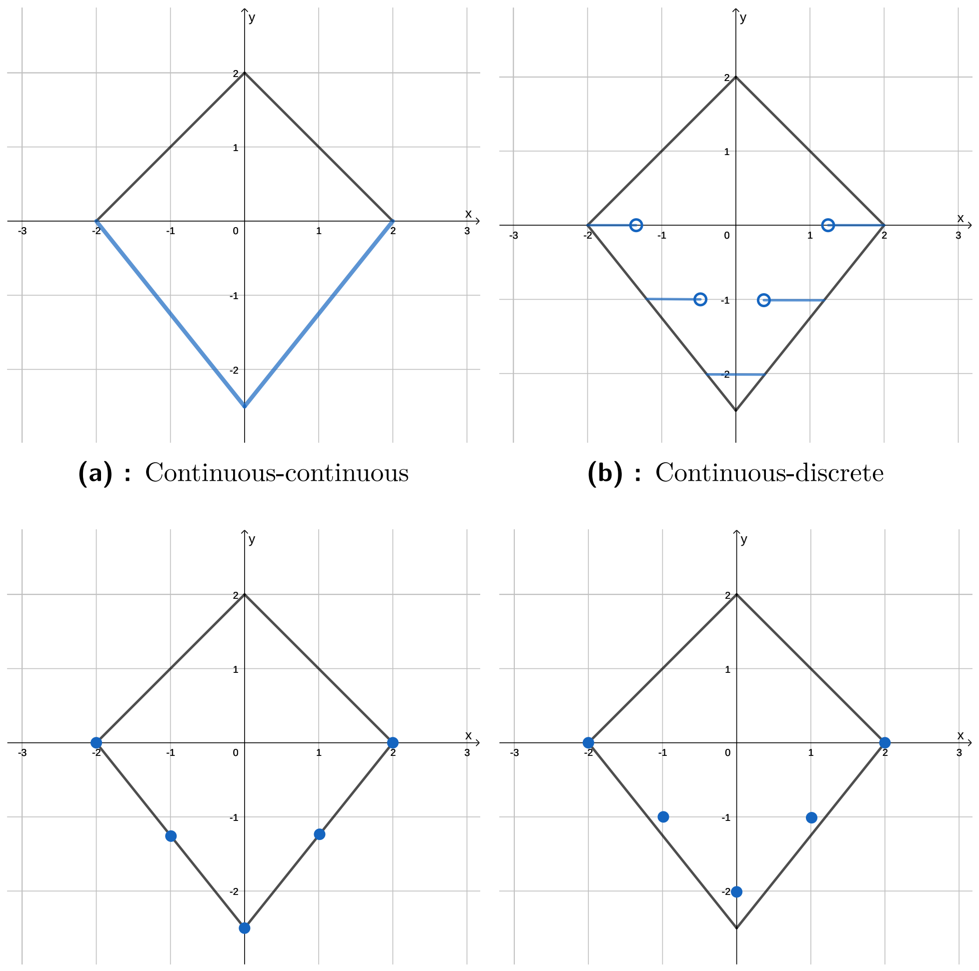
$$-x + y \leq 2 \tag{2.20}$$

$$5x - 4y \leq 10 \tag{2.21}$$

$$-5x - 4y \leq 10 \tag{2.22}$$

The inducible regions of the different bilevel problems with integer variables are shown in Figure 2.2, where blue areas denote the inducible region. The figure shows how introducing integrality changes the shape of the inducible region and how it can lead to a disconnected search space. For these reasons, the algorithms developed for solving the bilevel problems usually uses the specific features of each category.

Continuous-continuous category is studied the most as it allows the researchers to use the nice properties of the problems. Survey of methods for solving this type of problems is given in [10, 41]. One of the earliest work considering the mixed integer linear bilevel problem proposed a Branch and Bound scheme [33]. However, most of the fathoming rules cannot be used in these problems and the algorithm's nested structure is not scalable beyond few integer variables. In [12], algorithm utilizing Chvátal-Gomory cuts was proposed for solving the continuous-discrete problems. For problems containing integers in the upper level, a Benders decomposition algorithm [5] was presented in [39], with single-level reformulation of the slave problem. For discrete-continuous problems, it is also possible to use a single level reformulation by using Kerash-Kuhn-Tucker conditions or a duality based reformulation as given in [18].



(a) : Continuous-continuous

(b) : Continuous-discrete

(c) : Discrete-continuous

(d) : Discrete-discrete

Figure 2.2: Inducible regions of different linear bilevel problem categories. Blue lines/dots represent the inducible region of the problem.

Chapter 3

Problem Formulation

The considered problem tries to capture a real-world scenario, where project scheduling is not done on a single level, but instead multiple entities try to agree on a schedule feasible for all the parties involved. The problem considers two hierarchically organized entities in order to model the relations between a team leader representing the upper level and a project manager representing the lower level. The individual parties have the following objectives:

- The *team leader* considers staffing requirement and oversees hiring new people in case it is beneficial for resource levelling within reasonable costs, thus, the performance measure of this entity consists of a resource levelling and a resource hiring problem. In order to balance between the resource levelling and resource hiring costs, a weights representing the importance of the individual objectives are introduced – α for the resource levelling and β for the resource hiring.

- The *project manager's* objective is to make the project makespan as short as possible by minimizing the start time of the last dummy project activity. The makespan is its only interest and the agreement with the team leader can only be made when there is no possibility of coming up with a shorter schedule given the available workforce.

The notation used to describe the problem instance follows the notation presented in the previous chapter. Graph (N, A) represents the project network, where N denotes the set of activities and A the precedence constraints between activities. Set K contains the employee types and the considered set of discrete time steps is denoted as T. The duration of activity i is denoted as d_i with its requirement for personnel of type k denoted as r_{ik}.

The formulation of the problem is as follows:

$$\min F = \min \ \alpha \sum_{t=1}^{T-1} \sum_{k \in K} (j^+_{tk} + j^-_{tk}) + \beta \sum_{k \in K} y_k \tag{3.1}$$

$$\text{s.t. } y_k = \sum_{b \in B} 2^b y_{kb} \qquad \forall k \in K, \forall b \in B_k \tag{3.2}$$

$$y_k \leq R_k^{UB} \qquad \forall k \in K \tag{3.3}$$

$$y_k \in \mathbb{Z}_0^+ \qquad \forall k \in K \tag{3.4}$$

$$y_{kb} \in \{0,1\} \qquad \forall k \in K, \forall b \in B_k \tag{3.5}$$

$$\min \ f = \min \ \sum_{t \in T} v_{nt} t \tag{3.6}$$

$$\text{s.t.} \sum_{t \in T} v_{it} = 1 \qquad \forall i \in N \tag{3.7}$$

$$\sum_{t \in T} t v_{jt} - \sum_{t \in T} t v_{it} \geq d_i \qquad \forall (i,j) \in A \tag{3.8}$$

$$o_{k,t} \leq R_k + y_k \qquad \forall k \in K, \forall t \in T \tag{3.9}$$

$$o_{k,t} + j^+_{tk} - j^-_{tk} \leq o_{k,t+1} + M \sum_{t'=0}^{t} v_{nt'} \qquad \forall k \in K, \forall t \in T \tag{3.10}$$

$$M \sum_{t'=0}^{t} v_{nt'} + o_{k,t} + j^+_{tk} - j^-_{tk} \geq o_{k,t+1} \qquad \forall k \in K, \forall t \in T \tag{3.11}$$

$$v_{it} \in \{0,1\} \qquad \forall i \in N, \forall t \in T \tag{3.12}$$

$$j^+_{tk}, j^-_{tk} \in \mathbb{R}_0^+ \qquad \forall t \in T, \forall k \in K \tag{3.13}$$

Variable $o_{k,t}$, denoting the total resource k requirements at a time step t, is used to increase the readability of the problem formulation and is defined as follows:

$$o_{k,t} = \sum_{i \in N} \sum_{t'=t-d_i}^{t} r_{ik} v_{it'} \tag{3.14}$$

The leader controls integer variables y_k determining the number of additional personnel of type $k \in K$ hired. The resource levelling is represented by variables j^+_{tk} and j^-_{tk}, where the former represents jump at time t of resource k if the resource requirement of the next time step is higher, and the latter represents a jump in the opposite direction. In addition, the leader "controls" variables y_{kb}, which are directly determined from y_k as their binary representation. Apart from the additional personnel hired by the team leader, the team consists of stable employees represented by R_k for each resource type k. The maximum number of addition resource k units hired is given by R_k^{UB}, from which the set of bits B_k needed to represent y_k as a binary number is computed.

The follower controls binary variables v_{it} representing activity start at a given time for each activity i in project's activities and each time step t in the considered time period T. The follower also determines the resource jump variables j^+_{tk}, j^-_{tk} as those are directly determined from the schedule given by the follower.

The team leader's optimization task is given in Equation (3.1), where α determines the importance of resource levelling and β the importance of the additional workforce costs. Equation (3.2) creates the binary representation of variables y_k. Hired resources upper bound is given in Equation (3.3). Setting these bounds as tightly as possible may improve the algorithm runtime as it prunes the search space.

The follower's objective in Equation (3.6) consists of one term minimizing the project makespan. Constraints in Equation (3.7) make sure that all activities in the project start at some point. Compliance with the precedence relations of the project's activities is given in Equation (3.8). Satisfaction of the activity resource requirements at each discrete time step t is provided in Equation (3.9). Resource jump variables are determined in Equations (3.10) and (3.11). The M constant used in those constraints in combination with the minimization of the objective allows the jump variables to be set to 0 when the project is already finished, i.e., the project's last dummy activity has already started.

It can be easily seen that parameters α and β have a great influence on the problem structure. If α is set to a very low value compared to β, the problem basically collapses into a variation of the RACP, i.e., into finding a shortest possible feasible project schedule, where additional personnel is hired only when the current pool of workers is insufficient to create a feasible schedule. On the other hand, when α is set to a very large value compared to β, the problem changes into the resource levelling problem (RLP) where resources are to be set in a way so the project makespan of the follower's best response would equal to the one determined by the leader.

In conclusion, the problem of the team leader is to find the best schedule according to his or her combined performance measure of resource levelling and resource costs which at the same time has the shortest possible makespan across all other schedules achievable with the available resources.

Chapter 4

Proposed Algorithm

The algorithm for solving the stated problem needs to conduct a search in the inducible region in order to come with the optimal solution, but construction of inducible region beforehand is a hard task. For this reason, a different approach is fabricated. The core idea of the proposed approach is to search through the BLPP constraint region Ω and to continuously add constraints restricting the region so that the leader has to respect the requirements of the follower.

The idea of presenting additional constraints, also called *lazy constraints*, after an integral solution is found is already well-known and it is implemented in most modern solvers like Gurobi, IBM CPLEX and others. Example of the lazy constraints application can be found in [36], where the travelling salesman problem is solved with a subtour elimination, adding additional constraints that forbid cycles not spanning over the whole graph, if the solution does not cover all of the vertices.

The algorithm uses a mapping BR (best responses) to store the results of the computed lazy constraints for further use and speed-up of the algorithm. It maps a particular resource vectors $\mathbf{y^a}$ to the follower's best response objective value f_a^{FP}, i.e., it stores the best achievable project makespan for each vector of hired resources when the follower's problem is solved. The mapping is denoted as BR and is formalized as follows:

$$BR : \mathbf{y^a} \to f_a^{FP} \tag{4.1}$$

The overall scheme of the proposed algorithm is shown in Figure 4.1 with individual parts of the algorithm explained in detail in the next sections.

4.1 Initialization

The goal of the initialization process is to determine the considered time range in the optimization, i.e. the set of discrete time steps T and the upper

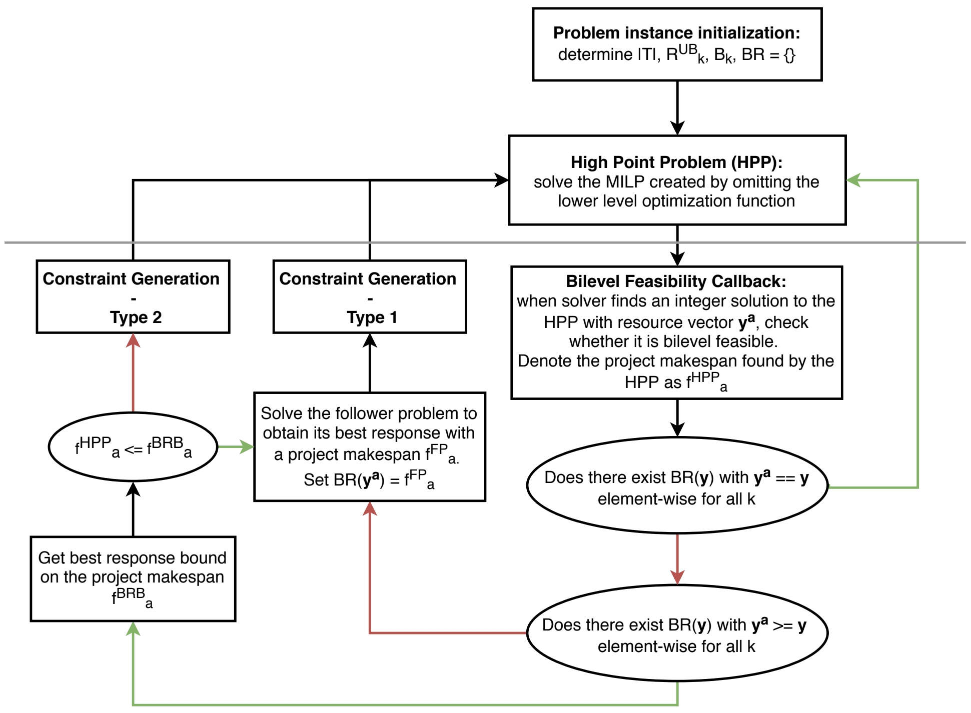

Problem instance initialization:
determine |T|, R^UB_k, B_k, BR = {}
High Point Problem (HPP):
solve the MILP created by omitting the lower level optimization function
Constraint Generation - Type 2
Constraint Generation - Type 1
Bilevel Feasibility Callback:
when solver finds an integer solution to the HPP with resource vector y^a, check whether it is bilevel feasible.
Denote the project makespan found by the HPP as f^HPP_a
f^HPP_a <= f^BRB_a
Solve the follower problem to obtain its best response with a project makespan f^FP_a.
Set BR(y^a) = f^FP_a
Does there exist BR(y) with y^a == y element-wise for all k
Get best response bound on the project makespan f^BRB_a
Does there exist BR(y) with y^a >= y element-wise for all k

Figure 4.1: Flow chart of the proposed Lazy constraints algorithm. Rectangular cells stand for algorithm steps, circular cells stand for flow control with green arrows denoting "Yes" and red arrows "No". The area below the gray line belongs to the callback execution flow.

bounds on the additional employees hired in order to determine the number of bits needed for their representation.

The minimal time step considered T_{min} can be found by solving the RCPSP for minimal makespan without the resource requirements taken into account. The maximal time step considered (T_{max}) can be found by solving a simple RCPSP given below in Equations (4.2)-(4.6). Considering longer time span is unnecessary as even though dealing with longer time range could result in better upper level objective, such solutions would never belong to the inducible region as they would not be bilevel feasible.

$$T_{max} = \min v_{nt}t \tag{4.2}$$

$$\text{s.t.} \sum_{t\in T} v_{it} = 1 \qquad \forall i \in N \tag{4.3}$$

$$\sum_{t\in T} tv_{jt} - \sum_{t\in T} tv_{it} \geq d_i \qquad \forall (i,j) \in A \tag{4.4}$$

$$\sum_{i\in N} \sum_{t'=t-d_i}^{t} r_{ik} v_{it'} \leq R_k \qquad \forall k \in K, \forall t \in T \tag{4.5}$$

$$v_{it} \in \{0,1\} \qquad \forall i \in N, \forall t \in T \tag{4.6}$$

The upper bound on the resources of type k hired is denoted as R_k^{UB}. The bound is computed from the project activities' resource requirements and it is needed for determining the number of bits in the binary representation of y_k. The resource upper bounds are computed by summing over the activities' resource demands as given below.

$$R_k^{UB} = \sum_{i \in N} r_{ik} \quad \forall k \in K \tag{4.7}$$

The numbers of bits $|B_k|$ required to represent the variable y_k is then equal to the number of bits needed to represent R_k^{UB}.

4.2 High point problem

The notion of using the *high point problem* (HPP) as a part of solving the bilevel problems was first presented in [6] and is also used in the general branch and bound scheme for mixed-integer bilevel linear programming in [33].

The HPP relaxes the original bilevel problem by omitting the lower level objective, thus, the HPP is an instance of a single level mixed-integer programming problem, where the leader gets to control both decision vectors. The optimal solution found for the HPP does not necessarily lie in the inducible region and to address this fact, a callback checking the bilevel feasibility is added, executed every time the solver finds an integer solution to the HPP. The callback is presented in the next section.

4.3 Bilevel feasibility callback

The goal of the callback is to check whether the solution found by the high point problem is valid in terms of bilevel feasibility. The callback extracts the project makespan from the high point integer solution denoted as f_a^{HPP} with a resource hire vector $\mathbf{y^a}$, and makes sure that there does not exist a schedule with shorter makespan with the provided resource limits. The shortest possible schedule is obtained by solving a simplified instance of the follower's RCPSP problem given below. Note that y_k^a used in the formulation now pose as constants, not variables.

$$\min v_{nt}t \tag{4.8}$$

$$\text{s.t.} \sum_{t \in T} v_{it} = 1 \quad \forall i \in N \tag{4.9}$$

$$\sum_{t \in T} t v_{jt} - \sum_{t \in T} t v_{it} \geq d_i \quad \forall (i,j) \in A \tag{4.10}$$

$$\sum_{i \in N} \sum_{t'=t-d_i}^{t} r_{ik} v_{it'} \leq R_k + y_k^a \quad \forall k \in K, \forall t \in T \tag{4.11}$$

$$v_{it} \in \{0,1\} \quad \forall i \in N, \forall t \in T \tag{4.12}$$

After obtaining the optimal follower's objective value f_a^{FP} for an integer solution a, which is the minimal project makespan for the particular vector resource hired by the leader $\mathbf{y^a}$, a verification that the project makespan of the HPP and the follower problem (FP) are equal needs to be done. It is not always necessary to solve the follower's problem as previously computed solutions can provide information about bilevel infeasibility. The best responses already solved are utilized providing a second type of lazy constraints without the computational burden of the first lazy constraint type that uses the optimal follower's solution.

Notation needed for the introduction of the lazy constraints' equations is given below. Let Y_{BR} be defined as a set of resource hire vectors, for which the follower's problem was solved to optimality and therefore, the best response of the follower is known. In other words, Y_{BR} denote the current domain of BR – the best responses mapping defined previously. Also, define $Y'(\mathbf{y^a})$ as the subset of the BR domain, where all the hired resources are element-wise greater or equal than $\mathbf{y^a}$ and at least one resource is strictly greater. The case when all of the hired resources are element-wise all equal should not occur as the lazy constraint with the given best response was already introduced to the problem.

$$Y'(\mathbf{y^a}) = \{\mathbf{y} \in Y_{BR} : y_k^a \geq y_k \; \forall k \in K \wedge \exists z \text{ such that } y_z^a > y_z\} \tag{4.13}$$

Then, if $Y'(\mathbf{y^a}) \neq \emptyset$, it is possible to define the *best response bound* (BRB) f_a^{BRB} for the vector $\mathbf{y^a}$ according to the subproblems solved so far as

$$f_a^{BRB} = \min_{\mathbf{y'} \in Y'(\mathbf{y^a})} BR(\mathbf{y'}) \tag{4.14}$$

Given the previous notation, the different cases of the callback procedure can finally be presented. The vector $\mathbf{y^a}$ and the project makespan of the HPP f_a^{HPP} are considered as the callback's input:

1. $Y'(\mathbf{y^a}) = \emptyset \vee f_a^{HPP} \leq f_a^{BRB}$:
   Solve the follower's problem to determine the shortest project makespan f_a^{FP}, store $\mathbf{y^a} \to f_a^{FP}$ to BR and add a lazy constraint of type 1 as given in Equation (4.15) to the HPP model.
2. $Y'(\mathbf{y^a}) \neq \emptyset \wedge f_a^{HPP} > f_a^{BRB}$:
   Add a lazy constraints of type 2 as given in Equation (4.16) to the HPP model.
3. Else:
   The solver found a bilevel feasible solution. Continue to solve the HPP without adding additional constraints.

The lazy constraints added to the problem use the binary representation of the leader's vector of resources hired and are formalized below. The first lazy constraint type is following:

$$1 - \left(\sum_{k \in K} \left(\sum_{b: y^a_{kb}=1} (y^a_{kb} - y_{kb}) + \sum_{b: y^a_{kb}=0} (y^a_{kb} + y_{kb}) \right) \right) \leq v_{n f^{FP}_a} \tag{4.15}$$

The second lazy constraint type is following:

$$1 - \sum_{k \in K} \sum_{b: y^a_{kb}=1} (y^a_{kb} - y_{kb}) \leq \sum_{t=0}^{t=f^{BRB}_a} v_{nt} \tag{4.16}$$

The lazy constraints of type 1 forces the model to discard all the solutions which does not comply with the best response of the follower for the given resource vector $\mathbf{y^a}$. The left side of Equation (4.15) is equal to zero if the binary representation of vector $\mathbf{y}$ is not equal to $\mathbf{y^a}$, not enforcing the project end at the specific time f^{FP}_a. When $\mathbf{y}$ equals to $\mathbf{y^a}$, the left side will be equal to one, forcing the project to end at time f^{FP}_a according to the optimal project makespan of the follower.

The principle behind lazy constraint of type 2 given in Equation (4.16) is similar, only now the enforced time is not exact, as with more resources shorter makespan can be achieved. This is represented by replacing the exact time variable with the sum at the right side of the equation. Also, the 0-bits are omitted from the left side of the equation as the constraint can cut solution with greater number of hired resources than $\mathbf{y^a}$.

The pseudo-code of the callback algorithm is given in Algorithm 1 and the procedure of finding the best possible bound according to the already solved follower's problems is given in Algorithm 2.

Algorithm 1 The callback used for checking bilevel feasibility of integral solutions and adding lazy constraints

```
1: function BilevelFeasibilityCallback($\mathbf{y^a}, f^{HPP}_a$)
2:     $f^{BRB}_a \leftarrow$ getBestResponseBound($\mathbf{y^a}$)
3:     if ($f^{BRB}_a$ is None or $f^{HPP}_a \leq f^{BRB}_a$) then
4:         $f^{FP}_a \leftarrow$ solveFollowerProblem($\mathbf{y^a}$)
5:         addLazyConstraintType1($f^{FP}_a$)
6:         BR.add($\mathbf{y^a} \rightarrow f^{FP}_a$)
7:     else if ($f^{HPP} > f^{BRB}_a$) then
8:         addLazyConstraintType2($f^{BRB}_a$)
9:     return
```

4.4 Algorithm summary

The proposed algorithm uses the single level high point problem formulation so it could be solved with modern-day solvers. By the solver's implicit

Algorithm 2 The algorithm obtaining best response bound from the stored solutions of the follower's problem

```
1: function GETBESTRESPONSEBOUND($\mathbf{y^a}$)
2:     $f_a^{BRB} \leftarrow$ None
3:     for all $\mathbf{y}$ in BR.domain() do        ▷ Go through all the best responses
4:         if $y_k^a \geq y_k$ for all $k \in K$ then
5:             if ($f_a^{BRB} = None$ or $BR(\mathbf{y}) < f_a^{BRB}$) then
6:                 $f_a^{BRB} \leftarrow BR(\mathbf{y})$
7:     return $f_a^{BRB}$
```

branch-and-bound algorithm along with the addition of the lazy constraints, the algorithm eventually constraints the solution space of the HPP such that it corresponds to the inducible region and arrives with the bilevel optimal solution. The algorithm utilizes the fact that the follower's problem optimization objective contains only one variable, therefore, with the binary representation of the leader's decision vector, it is possible to come up with effective lazy constraints restricting the search space to correspond to the inducible region of the bilevel problem.

Chapter 5

Comparison with Single-level Approaches

The problem presented in previous chapter is essentially concerning three objectives – resource levelling, workforce hiring and project makespan minimization – within the bilevel framework. This chapter tries to give a proof that the bilevel problem cannot be formulated with single-level approaches as those could yield different solutions in some of the instances. The single-level approaches examined are a sequential one, solving one objective at a time, dividing the problem into smaller well-defined ones, and an integrated one, trying to arrive at the same solution with the means of multi-objective formulation of the problem.

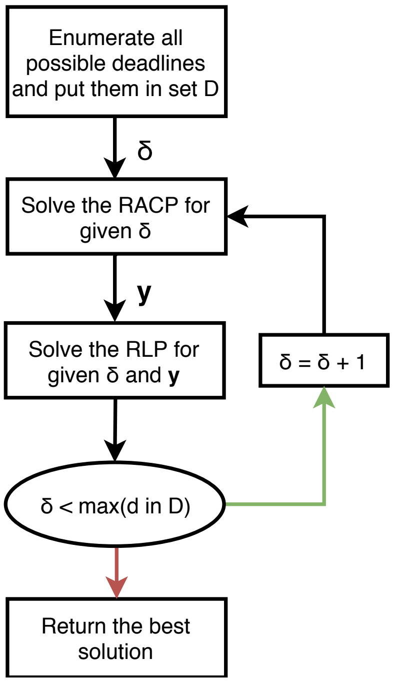
5.1 Sequential approach

The sequential approach consists of three steps. The first step enumerates all the possible makespans and stores them in a set D. The lowest makespan is computed as a minimal project makespan without taking the resource requirements into account and the highest makespan as a minimal project makespan with no additional resources hired. For each makespan δ, the other two steps are executed in a sequence with one of them being the resource-availability cost problem (RACP) solving the minimum number of employees needed to achieve the given makespan, and the second being the resource levelling problem (RLP) trying to shuffle the activity starts in order to minimize the number of resource jumps. The RLP and the RACP have the objective functions of $h^{SEQ,RLP}$, $h^{SEQ,RACP}$ respectively.

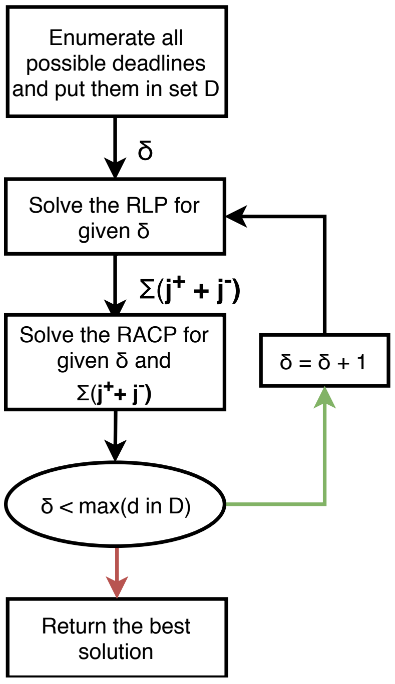
$$h^{SEQ,RACP} = \sum_{k \in K} y_k \tag{5.1}$$

$$h^{SEQ,RLP} = \sum_{t=1}^{T-1} \sum_{k \in K} (j^+_{tk} + j^-_{tk}) \tag{5.2}$$

These steps can be interchanged based on whether the importance is set on the resource levelling or workforce cost. The results computed for each


Enumerate all possible deadlines and put them in set D
δ
Solve the RACP for given δ
y
Solve the RLP for given δ and **y**
δ = δ + 1
δ < max(d in D)
Return the best solution

(a) : RACP before RLP


Enumerate all possible deadlines and put them in set D
δ
Solve the RLP for given δ
Σ(j⁺ + j⁻)
Solve the RACP for given δ and Σ(j⁺+ j⁻)
δ = δ + 1
δ < max(d in D)
Return the best solution

(b) : RLP before RACP

Figure 5.1: Schemas of the different sequential algorithms. The algorithm on the left solves the RACP first, the algorithm on the right solves the RLP first. Circular cells denotes flow control with green arrow denoting "Yes" and red arrows "No".

makespan are stored and after evaluating all the makespans, the best result is selected according to the leader's objective given in the bilevel approach (Equation (3.1)), denoted as F^{SEQ} with its optimal value of $F^{SEQ,*}$.

$$F = F^{SEQ} = \alpha \sum_{t=1}^{T-1} \sum_{k \in K} (j^+_{tk} + j^-_{tk}) + \beta \sum_{k \in K} y_k \tag{5.3}$$

The schemes of the sequential algorithms can be seen in Figure 5.1.

Counter-example giving a proof that the sequential approaches can arrive to different solution is easy to construct. A simple network of three activities (five if we also count the dummy activities) with a parallel structure is considered. All of the activities have duration $d_i = 1$ and workforce requirement $r_{i,1} = 1$ ($K = 1$). The number of available resources is $R_1 = 2$. Three makespans are possible in such a scenario with the schedules achieving them given below.

- $\delta = 1$: All activities start at time step 0, the resulting schedule has 0 jumps and 1 hired resource.
- $\delta = 2$: One activity starts at time step 0, the other two start at 1 or vice versa. Both of these cases arrive at 0 hired resource and 1 jump.

- $\delta = 3$: One activity starts at each time step achieving 0 jumps and 0 hired resources.

As there does not exist multiple options for each makespan that would arrive to different solutions when solving the separate problems (RLP, RCPSP), the order of the problems in the sequential approach does not matter. The results are summarized in Table 5.1.

Table 5.1: Results pointing out the difference between the sequential and bilevel approach. F^{SEQ} shows the value of the leader's objective and the last columns says, whether the schedule is bilevel feasible.

δ	$\sum_t(j_t^+ + j_t^-)$	y	F^{SEQ}	**Bilevel Feasible**
1	0	1	c	Yes
2	1	0	α	Yes
3	0	0	0	No

One of the points showing why the sequential approach fundamentally differs from the bilevel one can be seen from the table. The sequential approach always ends up with solution having $\delta = 3$, as it has no jumps and hires no additional resources, thus, has the minimal objective value for the leader (considering positive values of α and β). However, this solution does not belong to the inducible region as the makespan with no hired resources can be made shorter. This shows that the sequential approach may arrive to solutions that are not bilevel feasible, and therefore does not model the interaction between the two entities as the bilevel approach.

The second point proving the difference between the approaches can not be seen on the simple example presented. It is connected with the information and the objective value lost by addressing the objectives separately. If the RACP is solved first for each makespan, then the number of jumps can be high as the RACP does not consider them and in some instances it is beneficial to hire additional resources just to make the workforce utilization more leveled. The same principle applies when we switch the order of the problems in the sequence. By solving solely the resource levelling first, a schedule with a large number of resources that are not needed to achieve the given makespan may be hired.

In conclusion, the sequential approach differs from the bilevel one because it does not concern the bilevel feasibility of the schedules and because it may arrive at suboptimal solutions by solving one objective at a time. The gap between the objective value of the approaches can be arbitrary and depends on the order of the problems in the sequence, on the weights of the objectives in the bilevel approach and on the problem instance.

5.2 Multi-objective approach

Another approach representing the bilevel problem via combination of three objectives is the multi-objective single-level approach. The problem combines all of the three objectives (resource levelling, workforce hiring and makespan minimization) into one and assigns weights (α, β, γ) to each objective respectively. The objective of the integrated approach is given in Equation (5.4). The rest of the formulation follows the high point model excluding the constraints defining the binary variables.

$$\min h^{INT} = \min \alpha \sum_{t=1}^{T-1} \sum_{k \in K} (j_{tk}^{+} + j_{tk}^{-}) + \beta \sum_{k \in K} y_k + \gamma \sum_{t \in T} v_{nt} t \tag{5.4}$$

Let us denote the leader's objective function as F and F^{INT}, where the former represents the objective of the bilevel approach and the latter the objective of the integrated approach with the optimal solution values of F^* and $F^{INT,*}$, respectively.

$$F = F^{INT} = \alpha \sum_{t=1}^{T-1} \sum_{k \in K} (j_{tk}^{+} + j_{tk}^{-}) + \beta \sum_{k \in K} y_k \tag{5.5}$$

To show that the integrated approach differs from the bilevel one, the following statement needs to be contradicted by finding a counter-example:

> For every instance of the bilevel problem with α and β given by the bilevel problem, there exists a γ achieving the same schedule as the bilevel approach.

For the counter-example, it must hold that there does not exist γ such that the optimal solution of the integrated approach would be the same as the bilevel one. Let us denote the project makespan of the optimal solution of the bilevel approach as f^*. To find the appropriate value of γ arriving at the same makespan as the bilevel approach, a bisection method is used. The algorithm starts with an interval $[\gamma_{min}, \gamma_{max}]$ and sets the current value of γ as

$$\gamma = \frac{\gamma_{max} + \gamma_{min}}{2}. \tag{5.6}$$

After obtaining γ, the algorithm solves the integrated problem to optimality with a project makespan of the integrated solution denoted as $f^{INT,*}$. Three cases are then distinguished according to the difference between the makespans of the different approaches.

1. $f^{INT,*} < f^*$: The value of γ forced the schedule to have a shorter makespan than the bilevel program. Set $\gamma_{max} = \gamma$, compute new γ according to the updated interval and solve the integrated problem again.

2. $f^{INT,*} > f^*$: The value of γ forced the schedule to have a higher makespan than the bilevel program. Set $\gamma_{min} = \gamma$, obtain new γ according to the updated interval and solve the integrated problem again.

3. $f^{INT,*} = f^*$: The makespans are equal, if the leader's objectives are equal for both of the approaches ($F^* = F^{INT,*}$), then the current γ arrives at the same solution (or there exist a solution having the same makespan which is identical to the bilevel one and have the same optimal objective value). However, if the leader's objective values differs, the counter-example is found.

4. $(\gamma_{max} - \gamma_{min}) < \epsilon$: Algorithm stopping criterion for the cases, where there is no γ arriving at the same project's makespan as the bilevel problem.

The network of the counter-example found is given in Figure 5.2. The

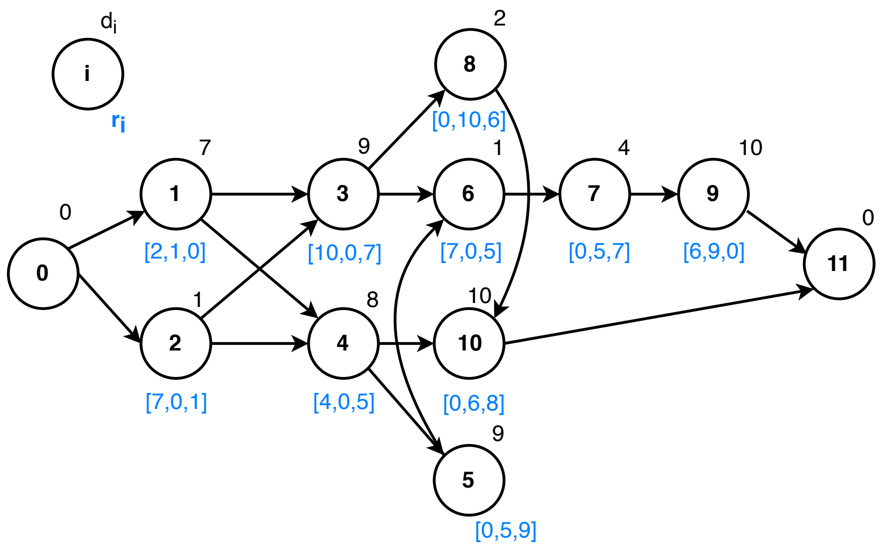
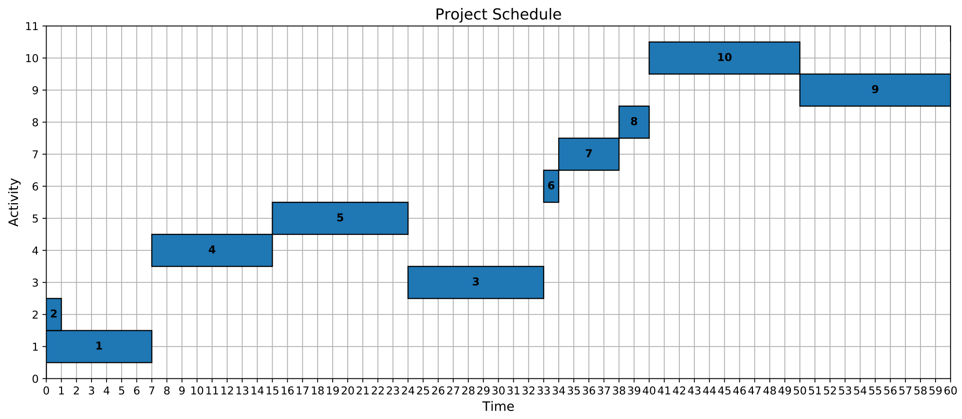
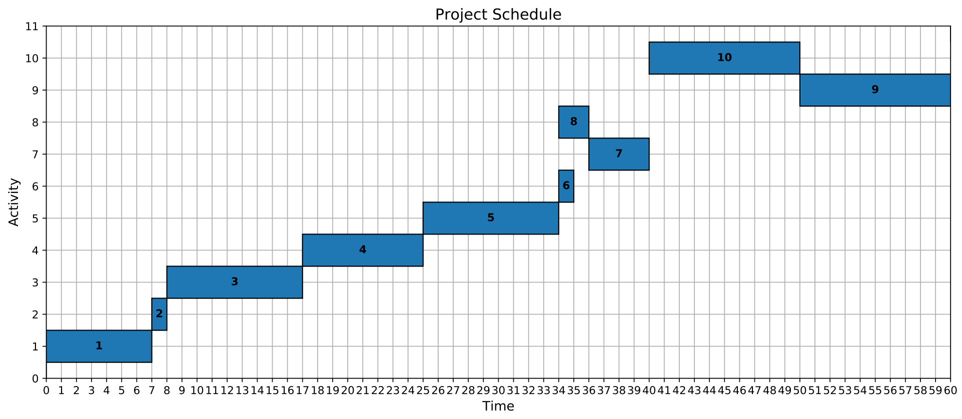


Figure 5.2: The problem instance used for proving the difference between the bilevel and integrated approach.

parameters of the bilevel problem are set to $\alpha = 25$, $\beta = 100$. The iterations of the integrated algorithm are shown in Table 5.2 along with the results of the bilevel approach. The optimal schedules for both approaches are displayed in Figure 5.3 The results show that even after achieving the same makespan the solution may not be identical. This is caused by the same principle which was presented in the sequential approaches – the integrated approach solution space is less restricted and does not need to comply with the bilevel feasibility. In the provided counter-example, the final step of the integrated approach hired an additional employee, however with the additional employee the project can be scheduled to a project makespan of 58, which is the reason why it is not the solution of the bilevel approach. Following observation can be made based on the counter-example.



(a) : Bilevel optimal schedule



(b) : Integrated optimal schedule

Figure 5.3: Optimal schedules of the bilevel and integrated approaches for the counter-example.

Table 5.2: Iterations of the integrated approach trying to find the same solution as the bilevel one. The algorithm stops after finding γ achieving the same project makespan as the bilevel approach.

Integrated approach					
γ	$f^{INT,*}$	$\sum_{t,k}(j^+_{tk} + j^-_{tk})$	$\mathbf{y}$	$F^{INT,*}$	$[\gamma_{min}, \gamma_{max}]$
950.00	39	98	$[0, 5, 6]$	3550	$[-100, 2000]$
425.00	39	98	$[0, 5, 6]$	3550	$[-100, 950]$
162.50	40	86	$[0, 5, 6]$	3250	$[-100, 425]$
31.25	59	88	$[0, 0, 1]$	2300	$[-100, 162.5]$
-34.38	60	88	$[0, 0, 1]$	2300	$[-100, 31.25]$
Bilevel approach					
	f^*	$\sum_{t,k}(j^+_{tk} + j^-_{tk})$	$\mathbf{y}$	F^*	
-	60	94	$[0, 0, 0]$	2350	-

Observation 5.1. Given an instance of the RCPSP and its solution by the bilevel approach with its final project makespan of f^* and the leader's objective value of its optimal solution F^*. If γ such that $f^{INT,*} = f^*$ exists, then $F^{INT,*} \leq F^*$.

Another experiment with the example aims to investigate whether the integrated approach can achieve arbitrary makespan just by setting the parameter γ, with α and β given by the bilevel approach. The experiment consists of iterating over all possible makespans and trying to reach the makespan with the same method used previously, only now the makespan is not set by the bilevel solution. The experiment shows that it not always possible to find γ achieving certain makespan as for some of the makespans the solution tends to oscillate between two different solutions – one achieving a longer makespan and other achieving a shorter makespan. The example of such behaviour is given in Table 5.3, where the network used previously is utilized and the algorithm tries to achieve a makespan of $\delta = 55$. Following observation can be made according to the results:

Observation 5.2. For a non-zero α and β and a makespan δ, there does not always exists γ such that the optimal solution of the integrated problem with makespan $f^{INT,*} = \delta$.

5.3 Comparison of all approaches

To achieve better insights into the differences of all the approaches, a final experiment comparing the properties of all the approaches is conducted. It consists of taking all the possible makespans of the example in Figure 5.2 and adding a constraint forcing the makespan for all the approaches, e.g., the project has to end precisely at the given makespan. This allows us to see the optimal solutions and feasibility per makespan per approach, allowing us to make further assumption about the different behaviours and properties of the approaches. The results displaying the leader's objective value of the optimal solution per makespan per approach are shown in Figure 5.4.

The figure allows to make another observation about the properties of the solution space of the approaches:

Observation 5.3. Existence of an integrated solution for makespan δ does not provide information about the existence of bilevel solution for that makespan and vice versa.

This means that there are makespans for which exists a bilevel solution but not an integrated one (e.g. $\delta = 53$ in Figure 5.4) and at the same time there

Table 5.3: Iterations of the integrated approach trying to achieve a makespan of $\delta = 55$. The makespan of the optimal solution is denoted as $f^{INT,*}$, the leader's objective value F and the objective value of the optimal solution for the integrated approach as $h^{INT,*}$.

γ	$f^{INT,*}$	$\sum_{t,k}(j_{tk}^{+} + j_{tk}^{-})$	$\sum_k y_k$	F^*	$h^{INT,*}$	$[\gamma_{min}, \gamma_{max}]$
0.00	59	1	88	2300	2300.00	[-10000, 10000]
5000.00	39	11	98	3550	198550.00	[0.0, 10000]
2500.00	39	11	98	3550	101050.00	[0.0, 5000.0]
1250.00	39	11	98	3550	52300.00	[0.0, 2500.0]
625.00	39	11	98	3550	27925.00	[0.0, 1250.0]
312.50	39	11	98	3550	15737.50	[0.0, 625.0]
156.25	40	11	86	3250	9500.00	[0.0, 312.5]
78.12	40	11	86	3250	6375.00	[0.0, 156.25]
39.06	51	5	82	2550	4542.19	[0.0, 78.125]
19.53	59	1	88	2300	3452.34	[0.0, 39.0625]
29.30	59	1	88	2300	4028.52	[19.531, 39.063]
34.18	51	5	82	2550	4293.16	[29.297, 39.063]
31.74	51	5	82	2550	4168.65	[29.297, 34.180]
30.52	59	1	88	2300	4100.54	[29.297, 31.738]
31.13	59	1	88	2300	4136.55	[30.518, 31.738]
31.43	51	5	82	2550	4153.09	[31.128, 31.738]
31.28	51	5	82	2550	4145.31	[31.128, 31.433]
31.20	59	1	88	2300	4141.05	[31.128, 31.281]
31.24	59	1	88	2300	4143.30	[31.204, 31.281]
31.26	51	5	82	2550	4144.33	[31.242, 31.281]
31.25	51	5	82	2550	4143.85	[31.242, 31.261]
31.25	59	1	88	2300	4143.58	[31.242, 31.252]
31.25	59	1	88	2300	4143.72	[31.247, 31.252]
31.25	51	5	82	2550	4143.79	[31.250, 31.252]

are makespans for which exists an integrated solution but not a bilevel one (e.g. $\delta = 59$ in Figure 5.4).

Also, it can be easily seen from the description of the approaches that by solving the problems in sequence in comparison to solving the problems combined in the integrated approach, following inequality must hold:

$$F^{SEQ,*} \geq F^{INT,*} \tag{5.7}$$

Furthermore, no relation can be established between F^* and $F^{SEQ,*}$ as the sequential approach can achieve a better objective value when arriving at the same solution as the integrated approach, but can also arrive at worse solution due to the sequentionality of the problems.

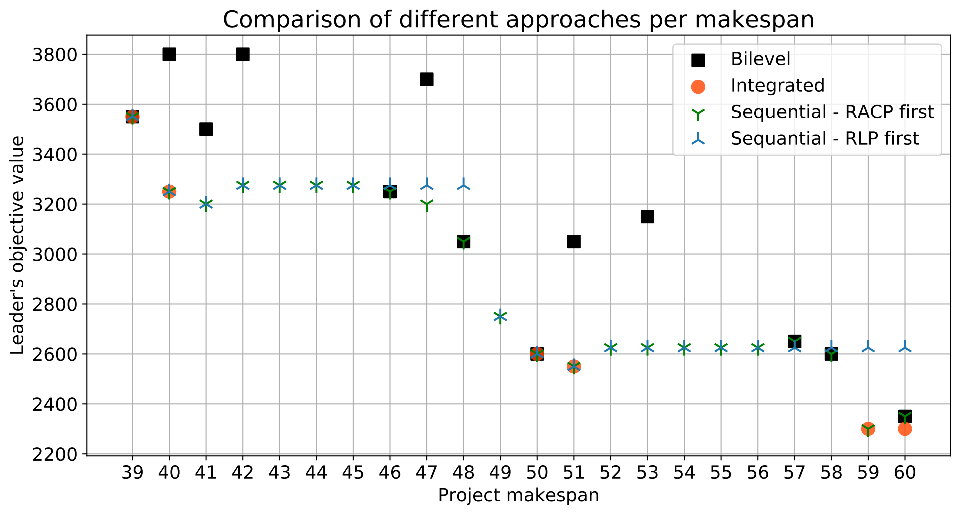


Figure 5.4: Leader's objective value of the different approaches for each makespan of the counter-example.

5.4 Summary

This section showed that the bilevel problem is unique and cannot be easily replaced by a single-level formulations which are most common when solving project scheduling problems. The single-level approaches fail to incorporate the best response of the follower, thus, simplifying the model and ignoring the interaction between different entities each pursuing its own interests. More experiments evaluating and comparing the approaches are provided in the next chapter.

Chapter 6

Experiments

This chapter presents the results of the algorithm's experimental evaluation and the influence of parameter setting on the computational performance, efficiency and solution quality.

The conducted experiments can be grouped into two categories – external and internal validation. The external validation tries to compare the presented approach with the current state-of-the-art algorithms. However, as the presented problem as well as the algorithm are novel and algorithms solving the problem cannot be found in the literature (the bilevel Branch and Bound scheme is not suitable for this amount of integer variables), the comparison will be done with the single-level approaches presented in the previous chapter. The internal validation focuses on evaluating the present algorithm in terms of computational performance, possible speed-ups, cuts efficiency and the influence of the instance parameters and objective weights on various metrics.

The parameters of the problem in the experiments are set as follows if not stated otherwise: the weight of the RLP is $\alpha = 25$ and the resource cost is $\beta = 100$. The solver used for the experiments is Gurobi 8.0.[1] including its Python wrapper. All the computations are done on a computer having Intel(R) Core(TM) i7-7600U CPU and 8 GB of memory.

6.1 Datasets

Two datasets are used to conduct the experiments. The first one is the well-known Patterson dataset [35] containing 110 problem instances. The second type of data instances used are generated by the RANGEN2 network generator presented in [46]. The RANGEN2 generates resource constrained project scheduling problems instances according to the input parameters, providing more control of the instances generated.

[1]`http://www.gurobi.com/`

6.1.1 RANGEN2

The parameters that are possible to use when generating new project instances are following:

- Number of activities $I1$.
- Series-Parallel indicator $I2 \in [0,1]$, where 0 represents all activities (not including the dummies) in parallel and 1 represents all activities in a sequence. The indicator is computed as follows:

  $$I2 = \frac{m-2}{n-3}, \tag{6.1}$$

  where n is the number of activities including the dummies and m is the progressive level of the end dummy activity. The progressive level PL_i of activity i is computed as

  $$PL_i = \begin{cases} 0 & \text{if } P_i = \emptyset \\ \max_{j \in P_i} PL_j + 1 & \text{else} \end{cases}, \tag{6.2}$$

  where P_i is the set of direct predecessors of activity i.
- Resource use (RU) specifying how many resource types each activity uses.
- Resource constrainedness (RC) specifying how much resources does an activity need in average with respect to the available resource pool.

The following notation is used to name the datasets generated by the RANGEN2: `RGxx_yy` is a dataset where `xx` stands for the number of activities not including the dummies, and `yy` denotes the value of the series-parallel indicator with which the network was generated.

6.1.2 Patterson dataset

Even though the Patterson dataset in not frequently used nowadays as the instances are too easy for solving the single-level problems, it is suitable for evaluation of the proposed algorithm as the bilevel approach is more computationally demanding.

The properties of the dataset are depicted in Figure 6.1. The figure shows that the instances are rather homogeneous, most of them having 20-30 activities, with majority using all of the resource types. Furthermore, most of the networks in the dataset are rather parallel with the series-parallel indicator being around 0.3.

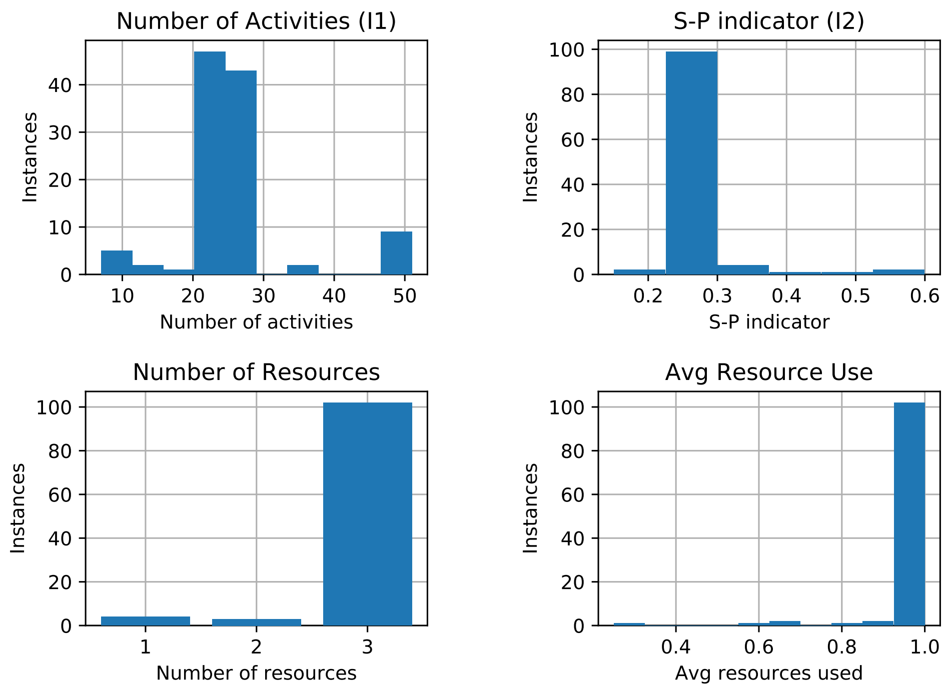


Figure 6.1: Patterson dataset properties

6.2 External validation

The experiments that belong to the external validation aim to compare the bilevel approach with other existing approaches in terms of both computational performance and solution quality. As there currently does not exist other method of solving the presented problem, the comparison is done with the sequential and integrated approaches presented in the previous chapter. The experiments are conducted on datasets: `RG10_03`, `RG10_05` and `RG10_07`, where each of those contain 100 problem instances. The bilevel algorithm with both types of lazy constraints is also referred to as the *baseline algorithm*.

The sequential approach assumes two algorithm variants – SEQ_RACP and SEQ_RLP – where the former solves first the resource availability cost before the resource levelling problem and the latter vice versa. In the integrated approach, the weight minimizing the project makespan is set to $\gamma = 0.1$, so the algorithm tries to minimize the leader's objective first and the makespan second, to imitate the bilevel scenario.

Computational comparison showing the runtime difference between the baseline algorithm and the other approaches can be seen in Figure 6.2. The results show that the integrated approach processing time is comparable to the bilevel problem. Also, the processing time of the sequential approaches differs based on which of the objectives is solved first – the results clearly show that solving the RACP task first for each deadline is faster than both the bilevel approach and the sequential approaches prioritizing the RLP.

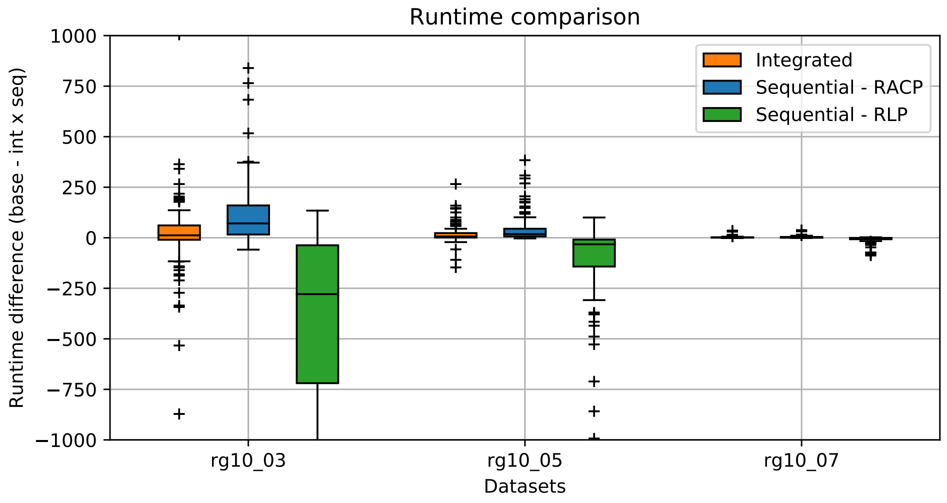


Figure 6.2: Runtime comparison of different approaches. The values shown are a subtraction of the other approaches runtime from the runtime of the baseline algorithm.

The comparison of the solution quality is depicted in Figure 6.3, which shows the percentage of the baseline objective value for each other approach, i.e., values above 100% denote worse objective value and values below 100% denotes improvement. The results show that the integrated approach always

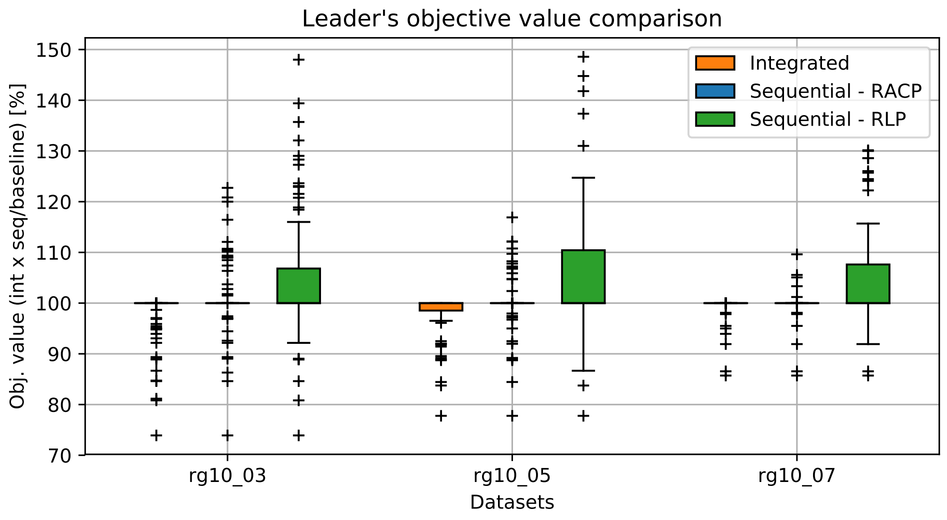


Figure 6.3: Comparison of leader's objective value for different approaches. The values are the ratio of the different approach and the baseline algorithm.

results in either the same or better objective value, further supporting the observations claimed in the previous chapter. Furthermore, it can be seen that the objective value of the sequential approach can be both better or worse than the bilevel one. The exact location of the bars of the sequential algorithms depends greatly on the α and β ratio. More detailed comparison of the solutions' components is given in Figure 6.4. The figure shows that the sequential approach with the RLP solved first tends to hire more resources than the bilevel approach, which can be seen in the left part of the figure.

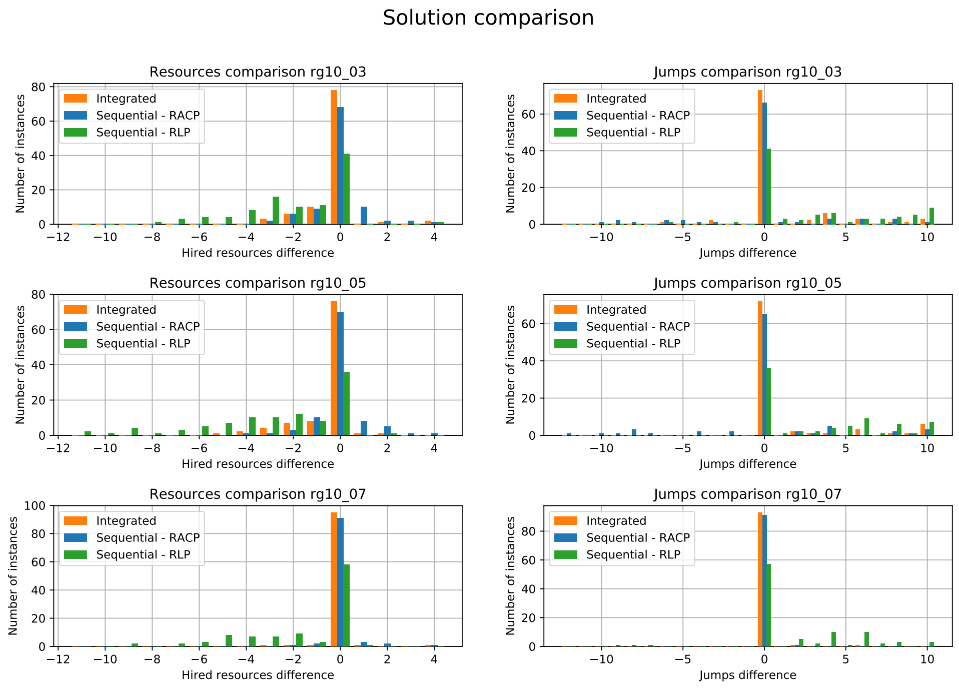


Figure 6.4: Solution components comparison. The values shown are a subtraction of the other other approaches results from the results of the baseline algorithm.

The comparison summary in terms of how many instances achieved a bilevel solution, how many arrived at the same solution as the bilevel one, etc. is given in Table 6.1. The table shows that within the weight setting used, the best approximation is given either by the integrated approach or by the sequential approach solving the RACP first. The integrated approach arrived at the same solution in more than 70 out of 100 cases in all of the considered datasets. The results also show, that bilevel feasibility and equality of the solutions of the bilevel and other approach increases with increasing $I2$ indicator, which is caused by the fact that the more the network resembles series, the less options there are for constructing the schedules, i.e., the schedule is more constrained by the precedence constraints.

6.3 Internal validation – speed-ups

This sections discusses the efficiency of the algorithm and its possible speed-ups. The efficiency of the second type of lazy constraint (LC2) is evaluated

Table 6.1: Summary of the results. Number of instances with the optimal solution being also bilevel feasible is given in column BF. Number of instances arriving at the same solution as the bilevel approach is given in the column SS. The comparison operators are used to compare the result of the different approaches with the bilevel solution, e.g., column Makespan $<$ denotes the number of instances where the sequential or integrated approach arrived at a result having smaller makespan than the bilevel solution.

Dataset	**Approach**	**Makespan**			**Objval**			**BF**	**SS**
		$=$	$<$	$>$	$=$	$<$	$>$		
rg10_03	INT	79	15	6	78	22	0	75	72
	SEQ_RACP	73	19	8	70	12	18	81	65
	SEQ_RLP	52	44	4	44	13	43	61	40
rg10_05	INT	78	15	7	74	26	0	75	71
	SEQ_RACP	75	16	9	73	13	14	81	64
	SEQ_RLP	48	45	7	37	21	42	49	32
rg10_07	INT	93	3	4	92	8	0	90	90
	SEQ_RACP	94	3	3	89	6	5	93	88
	SEQ_RLP	73	24	3	57	7	36	77	53

by running the algorithm without using it and comparing the processing time, total time spent in the lazy constraint callback and nodes explored in the search tree with the baseline algorithm. Another experiment discusses possible speed-up of the algorithm by utilizing a specific variable branching priority. The last experiment in this section tries to utilize additional information obtained from the network and constraint the variables further by using the earliest and latest start times of the activities that can be computed from the network.

The results of the experiments comparing the performance of the baseline algorithm and the bilevel algorithm not utilizing the second lazy constraint type are presented in Figure 6.5. The plot comparing the runtime shows that adding the LC2 makes the algorithm approximately twice as fast while also generally decreasing the number of explored nodes as can be seen in the plot showing the ratio of explored nodes. Furthermore, not utilizing the LC2 leads to increase in number of the LC1 generated by the algorithm and as a LC1 needs to solve an MIP model to be generated, it greatly increases the time spent in callbacks as can be seen in the plot showing the callback runtime ratio.

Another experiment tries to explore the influence of using variables branching priority in the algorithm. The branching strategy in the baseline algorithm is determined by the solver. Two other branching strategies are investigated, where one prioritizes the resource hiring $\mathbf{y}$ and the other prioritizes branching on the activity start variables $\mathbf{v}$. The results of the experiment are presented in Figure 6.6. The results show that enforcing the branching priority does not yield improvement as the results are either slightly worse or similar to using the implicit branching strategy determined by the solver.

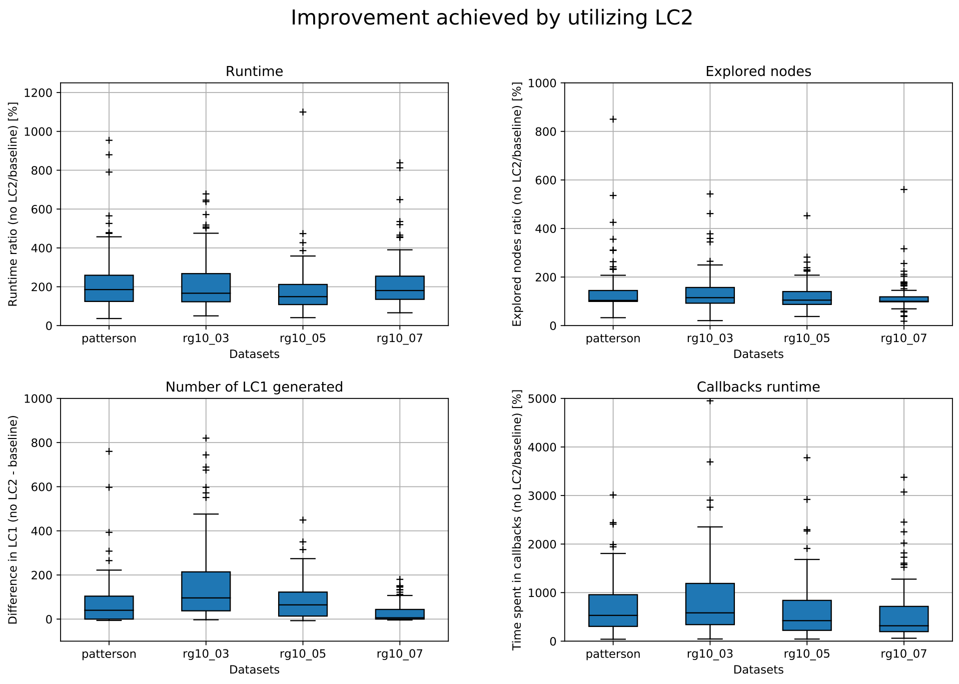


Figure 6.5: Results summarizing the improvements achieved by utilizing the second type of lazy constraint (LC2). The plots show a comparison of the baseline algorithm and the bilevel algorithm not using the LC2.

The last experiment in this section employs the information about earliest and latest start time (EST, LST) of each activity. The earliest start time of an activity is the earliest time activity can start considering only precedence constraints without the resource requirements. The latest start time of an activity is the latest time activity can start in order the project schedule would fit into the considered time period. The investigated speed-up scenario considers utilizing the EST and LST by setting

$$v_{it} = 0 \quad \text{if } t < EST_i \lor t > LST_i, \tag{6.3}$$

where EST_i is the earliest start time of activity i and LST_i is the latest start time of activity i. The results of comparison of the presented scenario with the baseline is given in Figure 6.7. The results show that the runtime have improved by using the additional activity start times information. Interestingly, the other indicators as the number of lazy constraints generated and number of nodes explored did not change much by employing the EST and LST, even though the runtime improved.

In conclusion, scenarios investigating possible speedups were conducted in this section along with an evaluation of the second lazy constraint type efficiency. The results show that utilizing the LC2 improves the computational time significantly, prioritizing certain variable does not yield visible improvements and that further runtime improvement can be achieved by utilizing the earliest and latest start time of the project's activities.

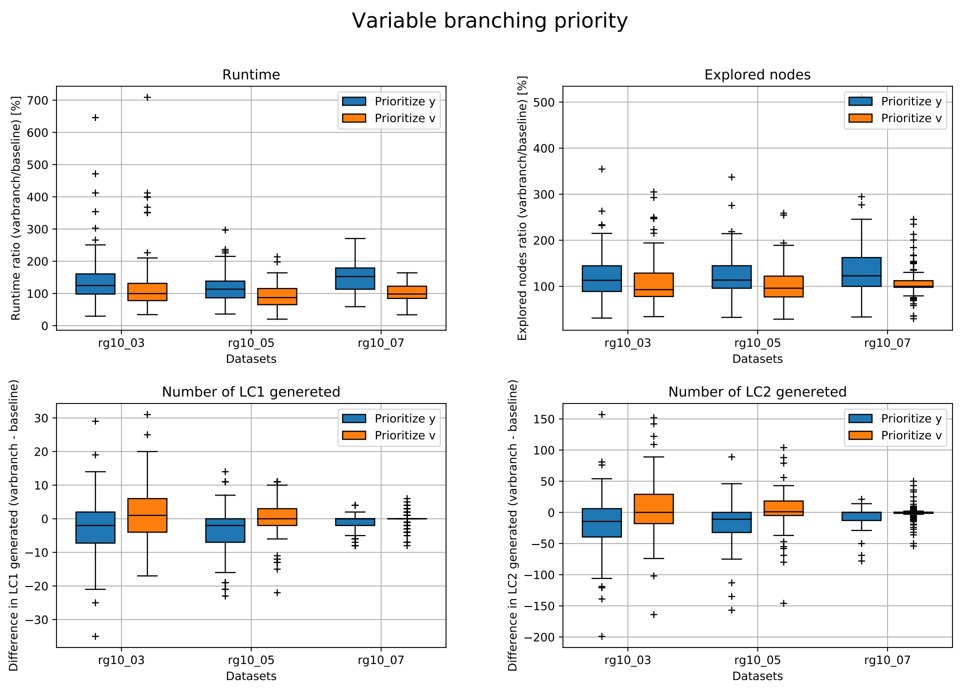


Figure 6.6: Comparison of different branching strategies with the baseline

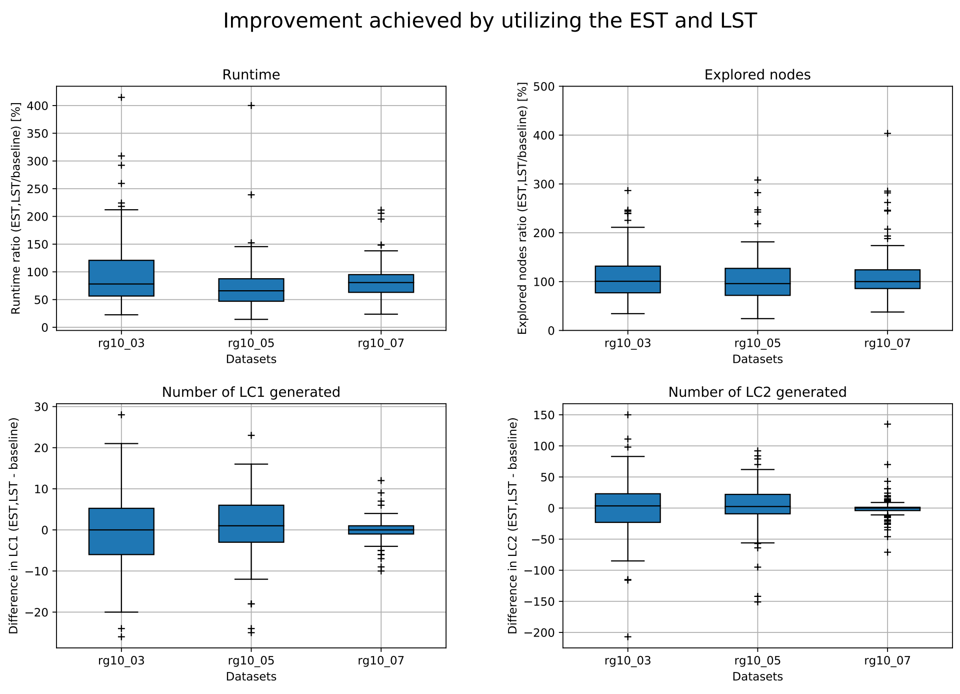


Figure 6.7: Comparison of a scenario using the EST and LST with the baseline

6.4 Internal validation – parameter setting

The algorithm has several parameters that can be set including the weights of the objectives α, β and the number of available resources R_k for each employee type. This section aims to investigate the influence of these parameters on the algorithm's performance. The dataset used for the evaluation is the `RG_10_05`.

The first experiment sets the objective weights α, β. The weights combination are drawn from a Cartesian product of $W \times W$, where

$$W = \{0, 1, 10, 100, 1000\} \tag{6.4}$$

The first element of the product is set as α and the second as β. Some combinations as $(0, 0)$ or $(10, 100)$ are omitted as they would not make sense or they are duplicates of previously explored combination (e.g. $(10, 100)$ would yield the same results as $(1, 10)$). The final combinations explored are as follows

$$\begin{aligned}(\alpha, \beta) \in \{&(0, 1), (1, 0), (1, 1), (1, 10), (1, 100),\\ &(1, 1000), (10, 1), (100, 1), (1000, 1)\}\end{aligned} \tag{6.5}$$

The results of the experiment is given in Figure 6.8. Note that all of the y-axis of the plots are logarithmic as otherwise some of the bars would not be readable. The figure shows that when the leader's objective does not

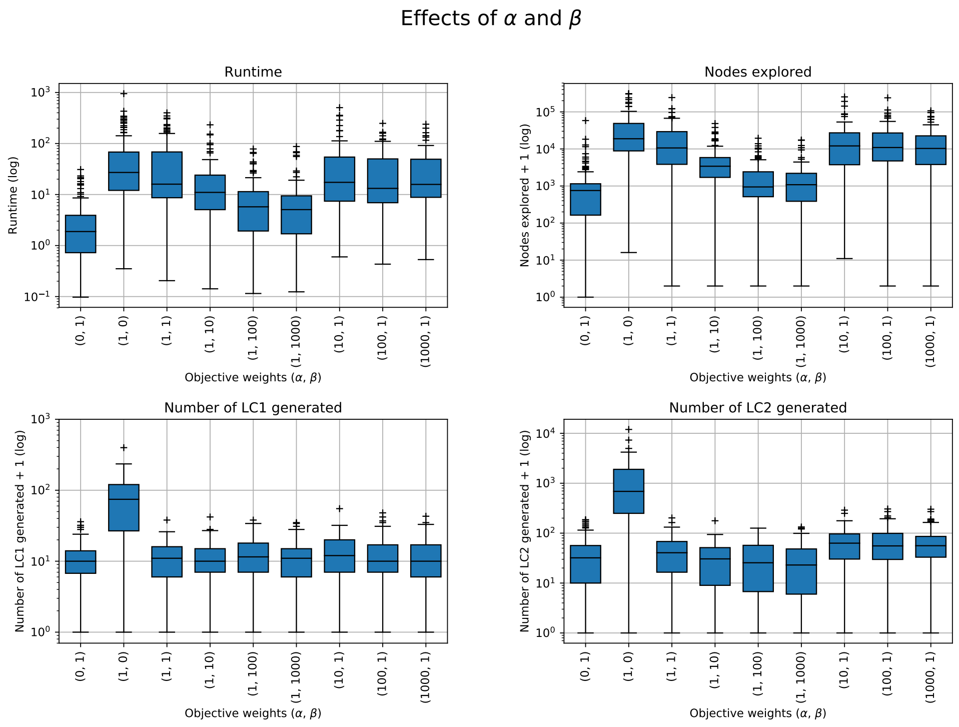


Figure 6.8: The algorithm's performance under various combinations of α and β

contain the resource levelling objective, i.e. $(\alpha, \beta) = (0, 1)$, the algorithm runs much faster as it does not have to deal with the non-regular objective. When the objective consists only of the RLP, i.e., $(\alpha, \beta) = (0, 1)$, the runtime is comparable to the other cases where the objective contains both problems. However, as this case contains no term trying to minimize the hired resources, the number of lazy constraints generated explodes compared to the other cases. The results also show, that the algorithm tends to run faster when the RACP weight α outweights the RLP weight β, even though the amount of generated lazy constraints generated is similar. This is caused by the solver being able to solve the high point problem more efficiently and cut off the search tree by its implicit procedures.

The second experiment sets the number of available workforce R_k for all resource types as R. The results are presented in Figure 6.9. The results

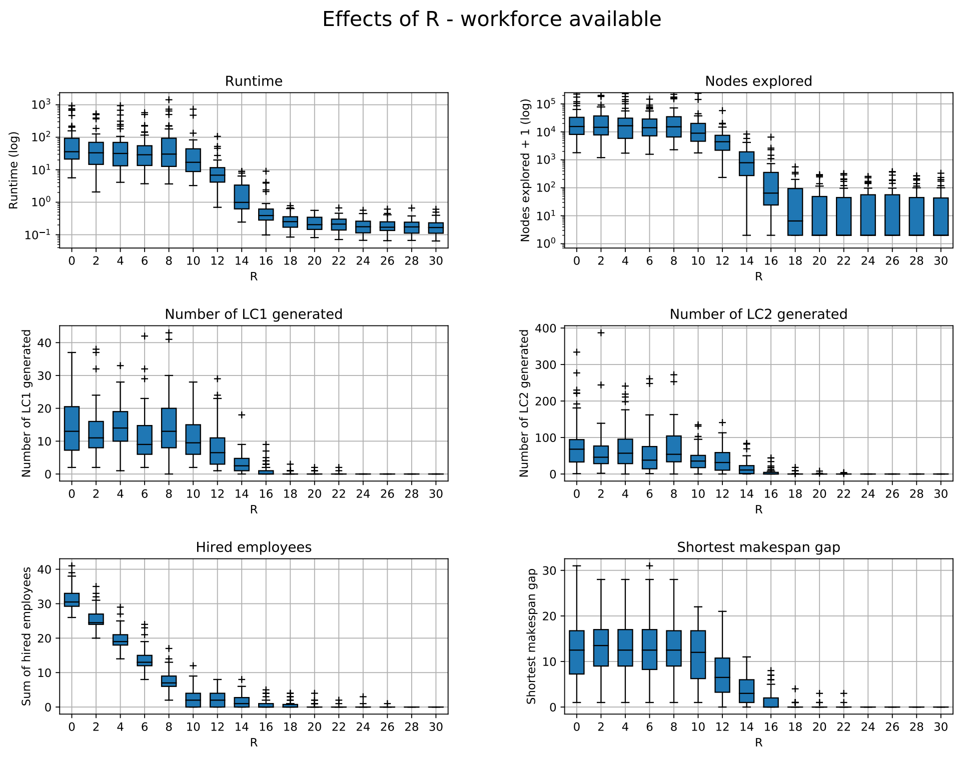


Figure 6.9: The algorithm's performance with different values of R

show that the performance is similar until reaching a threshold of $R = 10$, the runtime then drops with increasing value of R until reaching a second threshold $R = 18$. Increasing the value of R above the second threshold does not yield significant performance improvement. The results also show that no additional employees are hired in all of the instances when $R \geq 28$ and the algorithm always arrives at the shortest makespan possible when $R \geq 24$. Note that the results highly depend on the network generator parameters like the resource use, resource constrainedness and the objective weights, therefore, the results cannot be generalized.

6.5 Internal validation – problem instances

The last experiments section investigates the influence of the network parameters on the algorithm's performance. The examined parameters are the number of activities, number of resource types used in the project scheduling and value of the series-parallel indicator of the network.

The number of activities were evaluated on the Patterson dataset with the results presented in Figure 6.10. The plot shows that the activity number

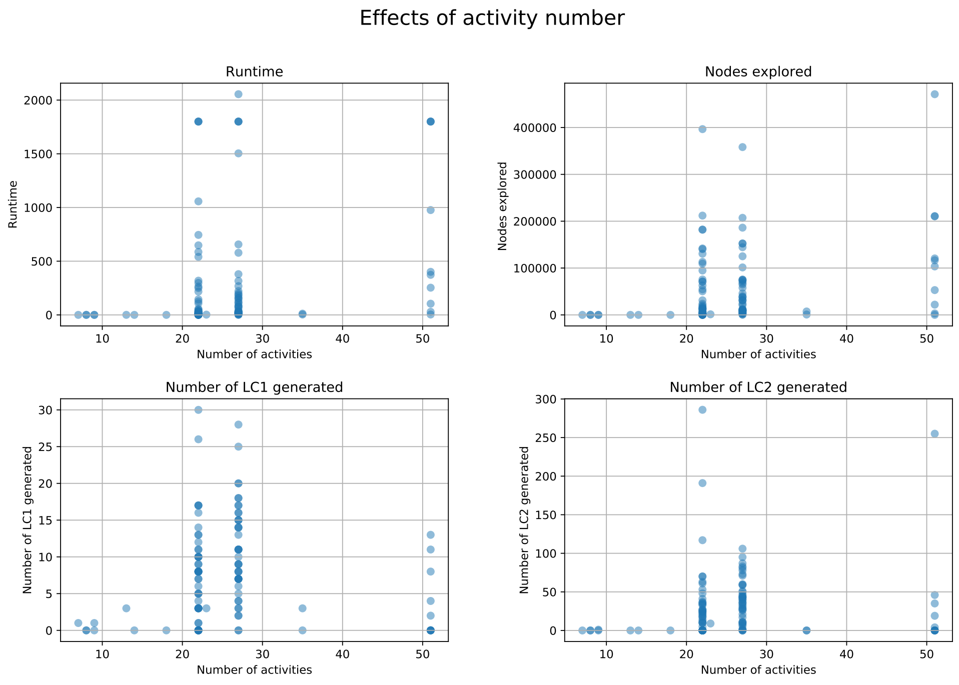


Figure 6.10: The algorithm's performance with different number of activities

is not the most important factor determining the computational speed as the results show no visible correlations between the number of activities and the performance indicators. Furthermore, running the experiments with RANGEN2 generated instances having more than 20 activities resulted in reaching the time limit in most of them. This further supports the claim that the complexity is not in the activity number as the algorithm could solve problems with more than 50 activities in the Patterson dataset.

Another experiment evaluated the effect of the series-parallel indicator with the results depicted in Figure 6.11. The results clearly show that with increasing parallelness of the network (lower values of the S-P indicator) the solution becomes harder to be found. This follows the fact that there are more options how to construct the schedule when the network resembles a parallel one more.

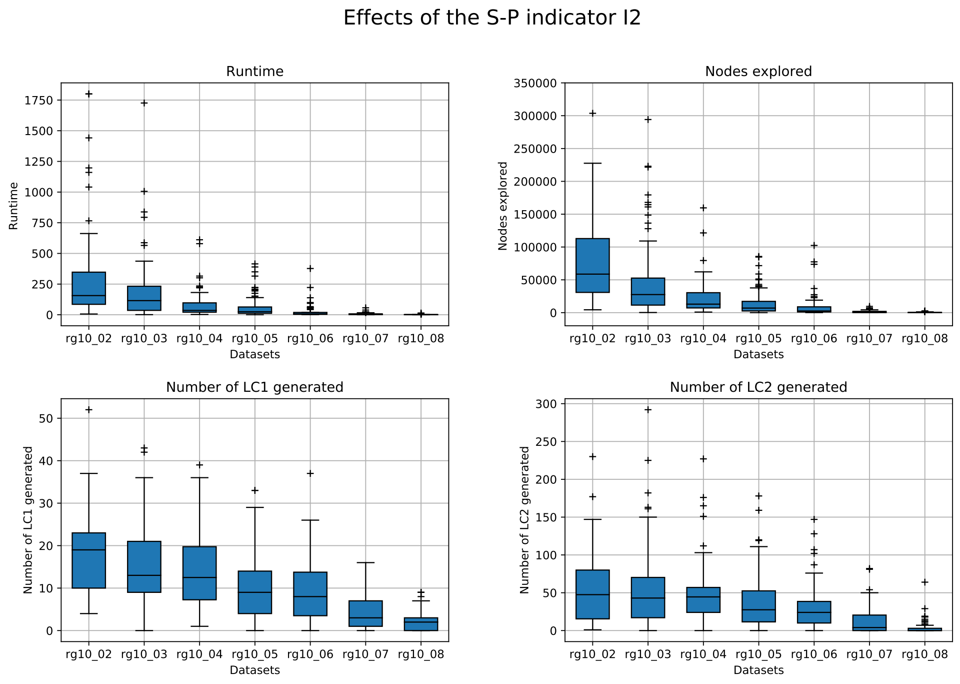


Figure 6.11: The algorithm's performance with different values of the series-parallel indicator

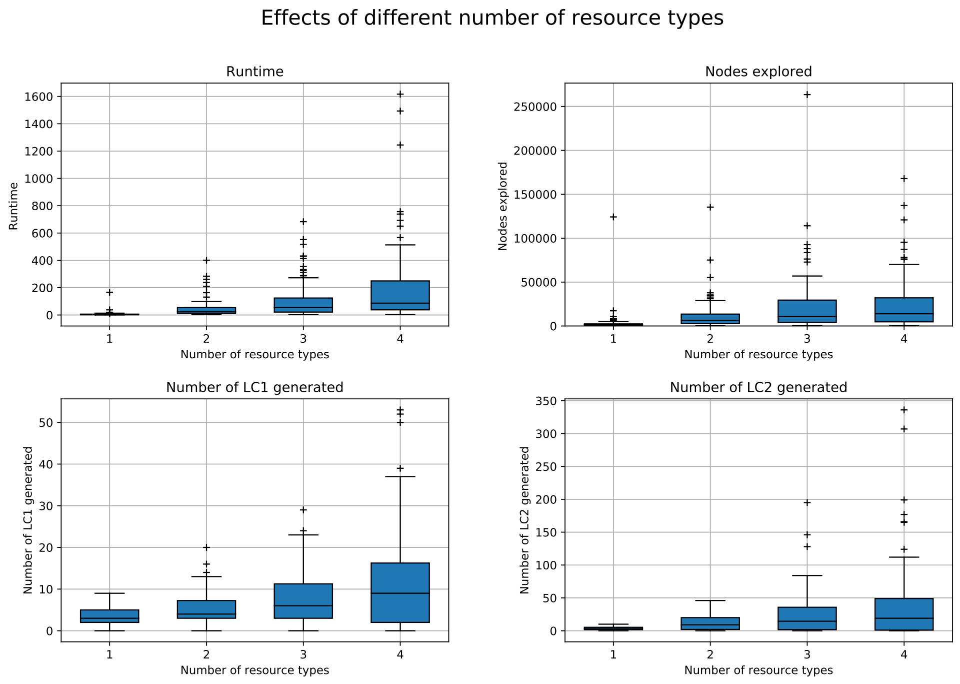


Figure 6.12: The algorithm's performance with different number of resource types

The last experiments explores the influence of the number of resource types used in the instances. The dataset used in the scenario is the `RG10_05` with the S-P indicator set to $I2 = 0.5$ and various resource types number. The outputs of the algorithm are shown in Figure 6.12 and shows that the complexity of finding the solution increases with increasing the number of resource types used. This follows the common-sense reasoning as increasing the number of resource types increases the problem size (the number of variables used).

6.6 Summary

Various experiments exploring the algorithm's properties were conducted and presented in this chapter. The experiments compared the bilevel algorithm with single-level solutions to provide idea on how much the bilevel problem differs from the single-level ones. Possible speed-up techniques were evaluated and showed that the second type of lazy constraint doubles the computational speed of the algorithm proving the efficiency of the constraint. Changing the variable branching strategy did not improve the algorithm's computational time and some improvement can be achieved by employing additional information about the network structure (the earliest and latest start time of activities). Further experiments shown the influence of various parameters setting of the algorithm and of the project network instances on the overall performance.

Chapter 7

Conclusion and Future Work

In this thesis, a model representing a decision making process within a context of resource contrained project scheduling in a hierarchically organized team is proposed. The model represents two entities – a team leader who is in charge of hiring additional employees and at the same time tries to make the resource usage as levelled as possible, and a project manager trying to minimize the project makespan. A bilevel programming formulation is proposed to model the interaction between the distinct entities pursuing their individual goals. The bilevel model puts the objective of the project manager to the constraints of the team leader's problem, creating a novel point of view to the project scheduling problem.

An algorithm based on lazy constraints generation is proposed to solve the devised model. The algorithm solves the high-point problem, while executing callbacks on the discovered integer solutions. The callback checks whether the project schedule complies with the requirements on the makespan introduced by the project manager. If it is possible to make the project makespan shorter with the employees hired by an integer solution of the high point problem, a lazy constraint is added to cut the solution. Thus, the algorithm always arrives at a solution satisfying both entities.

A comparison with sequential and integrated single-level approaches is made to show that the bilevel formulations cannot be easily replaced by a single-level algorithm and that the proposed problem is unique. Furthermore, experiments evaluating and benchmarking the algorithm are conducted to show the performance and properties of the algorithm.

7.1 Future work

The application of the bilevel optimization techniques introduces a new point of view on the project scheduling problems as it has not been used before. The idea can be applied further to other problems in operation research, where the possible interaction between entities pursuing different goals have been omitted so far and the problems were simplified as a single-level problems.

Regarding the algorithm, a lazy constraints generation has not yet been utilized in solving integer bilevel problems. Therefore, the algorithm should be further examined on other problems to see whether it is possible to generalize it.

Bibliography

[1] Alessandro Agnetis, Cyril Briand, Jean-Charles Billaut, and Přemysl Šůcha. Nash equilibria for the multi-agent project scheduling problem with controllable processing times. *Journal of Scheduling*, 2015.

[2] Bilal M. Ayyub and Achintya Haldar. Project scheduling using fuzzy set concepts. *Journal of Construction Engineering and Management - ASCE*, 1984.

[3] Cynthia Barnhart, Ellis L. Johnson, George L. Nemhauser, Martin W. P. Savelsbergh, and Pamela H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Oper. Res.*, 1998.

[4] Odile Bellenguez-Morineau and Emmanuel Neron. A branch-and-bound method for solving multi-skill project scheduling problem. *RAIRO - Operations Research*, 2007.

[5] Jacques F Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische mathematik*, 1962.

[6] Wayne F. Bialas and Mark H. Karwan. Two-level linear programming. *Management Science*, 1984.

[7] Dušan Bobera. Project management organization. *Management information systems*, 2008.

[8] Peter Brucker, Andreas Drexl, Rolf Möhring, Klaus Neumann, and Erwin Pesch. Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research*, 1999.

[9] Y. Chen and M. Florian. The nonlinear bilevel programming problem: formulations, regularity and optimality conditions. *Optimization*, 1995.

[10] Benoît Colson, Patrice Marcotte, and Gilles Savard. An overview of bilevel optimization. *Annals of Operations Research*, 2007.

[11] Erik L. Demeulemeester and Willy. Herroelen. *Project scheduling : a research handbook*. International series in operations research & management science ; 49. Boston : Kluwer Academic Publishers, 2002.

[12] S Dempe. Discrete bilevel optimization problems. Technical report, Inst. für Wirtschaftsinformatik, 2001.

[13] M.S. Dobson. *The Triple Constraints in Project Management*. Project Management Essential Library. Management Concepts Press, 2004.

[14] Salah E. Elmaghraby and Jerzy Kamburowski. The analysis of activity networks under generalized precedence relations (gprs). *Management Science*, 1992.

[15] S.E. Elmaghraby. *Activity Networks: Project Planning and Control by Network Models*. Wiley, 1977.

[16] International Organization for Standardization. Quality management systems – fundamentals and vocabulary. Standard, Geneva, CH, September 2015.

[17] Lu Gan and Jiuping Xu. Control risk for multimode resource-constrained project scheduling problems under hybrid uncertainty. *Journal of Management in Engineering*, 2015.

[18] Pablo Garcia-Herreros, Lei Zhang, Pratik Misra, Erdem Arslan, Sanjay Mehta, and Ignacio E. Grossmann. Mixed-integer bilevel optimization for capacity planning with rational markets. *Computers and Chemical Engineering*, 2016.

[19] Zeynep H. Gümüş and Christodoulos A. Floudas. Global optimization of nonlinear bilevel programming problems. *Journal of Global Optimization*, 2001.

[20] Maciej Hapke, Andrzej Jaszkiewicz, and Roman Slowinski. Fuzzy project scheduling system for software development. *Fuzzy Sets and Systems*, 1994.

[21] Sönke Hartmann. A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics (NRL)*, 1998.

[22] Sönke Hartmann and Dirk Briskorn. A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, 2010.

[23] Willy Herroelen, Erik Demeulemeester, and Bert De Reyck. *A Classification Scheme for Project Scheduling*. Springer US, 1999.

[24] Willy Herroelen, Bert De Reyck, and Erik Demeulemeester. Resource-constrained project scheduling: A survey of recent developments. *Computers & Operations Research*, 1998.

[25] Rainer Kolisch and Sönke Hartmann. Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European journal of operational research*, 2006.

[26] Rainer Kolisch and Sönke Hartmann. Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European Journal of Operational Research*, 2006.

[27] Charles D Kolstad. A review of the literature on bi-level mathematical programming. Technical report, Los Alamos National Laboratory Los Alamos, NM, 1985.

[28] Pieter Leyman and Mario Vanhoucke. Payment models and net present value optimization for resource-constrained project scheduling. *Computers & Industrial Engineering*, 2016.

[29] Broos Maenhout and Mario Vanhoucke. An exact algorithm for an integrated project staffing problem with a homogeneous workforce. *Journal of Scheduling*, 2016.

[30] Broos Maenhout and Mario Vanhoucke. A resource type analysis of the integrated project scheduling and personnel staffing problem. *ANNALS OF OPERATIONS RESEARCH*, 2017.

[31] D. G. Malcolm, J. H. Roseboom, C. E. Clark, and W. Fazar. Application of a technique for research and development program evaluation. *Operations Research*, 1959.

[32] Jorge Jose de Magalhaes Mendes, José Fernando Gonçalves, and Mauricio GC Resende. A random key based genetic algorithm for the resource constrained project scheduling problem. *Computers & Operations Research*, 2009.

[33] James T. Moore and Jonathan F. Bard. The mixed integer linear bilevel programming problem. *Operations Research*, 1990.

[34] Linet Ozdamar and Gunduz Olosoy. A survey on the resource-constrained project scheduling problem. *IIE Transactions*, 1995.

[35] James H. Patterson. A comparison of exact approaches for solving the multiple constrained resource, project scheduling problem. *Management Science*, 1984.

[36] Ulrich Pferschy and Rostislav Staněk. Generating subtour elimination constraints for the tsp from pure integer solutions. *Central European Journal of Operations Research*, 2017.

[37] A. Alan B. Pritsker, Lawrence J. Waiters, and Philip M. Wolfe. Multiproject scheduling with limited resources: A zero-one programming approach. *Management Science*, 1969.

[38] Bert Reyck and Willy Herroelen. An optimal procedure for the resource-constrained project scheduling problem with discounted cash flows and generalized precedence relations. *Computers & Operations Research*, 1998.

[39] G. K. Saharidis and M. G. Ierapetritou. Resolution method for mixed integer bi-level linear problems based on decomposition technique. *Journal of Global Optimization*, 2009.

[40] Gilles Savard and Jacques Gauvin. The steepest descent direction for the nonlinear bilevel programming problem. *Operations Research Letters*, 1994.

[41] A. Sinha, P. Malo, and K. Deb. A review on bilevel optimization: From classical to evolutionary approaches and applications. *IEEE Transactions on Evolutionary Computation*, 2018.

[42] Vincent Van Peteghem and Mario Vanhoucke. A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research*, 2010.

[43] Vincent Van Peteghem and Mario Vanhoucke. Heuristic methods for the resource availability cost problem. In C Schwindt and J Zimmermann, editors, *Handbook on project management and scheduling*, International Handbooks on Information Systems, pages 339–359. Springer International Publishing Switzerland, 2015.

[44] Mario Vanhoucke. *Project management with dynamic scheduling: Baseline scheduling, risk analysis and project control.* Springer, 2012.

[45] Mario Vanhoucke and José Coelho. An approach using sat solvers for the rcpsp with logical constraints. *European Journal of Operational Research*, 2016.

[46] Mario Vanhoucke, José Coelho, Dieter Debels, Broos Maenhout, and Luís V. Tavares. An evaluation of the adequacy of project network generators with systematically sampled networks. *European Journal of Operational Research*, 2008.

[47] Mario Vanhoucke, Erik Demeulemeester, and Willy Herroelen. On maximizing the net present value of a project under renewable resource constraints. *Management Science*, 2001.

[48] L. Vicente, G. Savard, and J. Judice. Discrete linear bilevel programming problem. *Journal of Optimization Theory and Applications*, 1996.

[49] H. von Stackelberg, D. Bazin, R. Hill, and L. Urch. *Market Structure and Equilibrium.* Springer Berlin Heidelberg, 2010.

[50] H. von Stackelberg and A. Peacock. *The theory of the market economy*. Hodge, 1952.

[51] J. Xu, Y. Ma, and Z. Xu. A bilevel model for project scheduling in a fuzzy random environment. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2015.

Appendix A

CD Contents

- `/thesis.pdf` – The thesis in a pdf format
- `/specification.pdf` – The thesis specification
- `/src/` – Folder containing the Python source codes
- `/src/results/` – Folder containing the results of the conducted experiments
- `/src/datasets/` – Folder containing the datasets used for experiments

Appendix B

Abbreviations

RCPSP	resource-constrained scheduling problem
RACP	resource availability cost problem
RCPSPDC	resource-constrained scheduling problem with discounted cashflow
BLPP	bilevel linear programming problem
IR	inducible region
RLP	resource leveling problem
HPP	high point problem
FP	folower's problem
LC1, LC2	lazy constraint of type 1, 2, respectively
MIP	mixed-integer program
EST, LST	earliest, latest start time, respectively