SAPIENZA
UNIVERSITÀ DI ROMA

Integer Bilevel Linear Programming Problems: New Results and Applications

Scuola di Dottorato in Scienza e Tecnologia dell'Informazione delle Comunicazioni

Dottorato di Ricerca in Ricerca Operativa – XXVI Ciclo

Candidate

Renato Mari
ID number 1381561

Thesis Advisor

Prof. Massimiliano Caramia

A thesis submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Operation Research

Thesis defended on 16^{th} May 2014
in front of a Board of Examiners composed by:

Prof. Luca Zanni (chairman)

Prof. Andrea Lodi

Prof. Maria Grazie Scutellà

Integer Bilevel Linear Programming Problems: New Results and Applications
Ph.D. thesis. Sapienza – University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: renato.mari@uniroma1.it

Dedicated to
my family

Abstract

Bilevel programming problems have received an increasing attention in the last two decades. Many papers, surveys and research activities are focused on this topic because, on one side it represents a novel and interesting approach to model a wide range of real life applications, but on the other side it is still a very challenging problem for which only a small subclass of instances with particular properties can be efficiently solved. In literature most of the results are manly theoretical and there is a lack of efficient algorithms able to solve general instances of these problems.

Bilevel programming problems were historically introduced to model defense problems and optimization problems in multilevel organization. The general framework of a multilevel programming problem is the presence of multiple decision makers with conflicting objectives in a no cooperative scenario in which the strategies and the actions of one decision maker strictly depend on the other decision makers involved and, unlike classical multiobjective optimization problems, there is an explicit hierarchical relationship among them. Modelling these kind of situations by means of a classical single level optimization problem has the flaw not to capture the conflicting nature of objectives involved and may not be realistic. This explains the increasing interest of the scientific community toward this class of problems.

In this dissertation we focus on bilevel programming problems, multilevel problems in which only two decision makers are considered. For the sake of clearness, the first player is called *leader* and the second player is called *follower*. The leader is the first decision maker who plays knowing the rational reaction of the follower and taking it into account. The follower reacts to the leader's decision trying to optimize his objective function. The resulting formulation is an optimization problem in which the leader's feasible set is defined by both a set of constraints and the follower's optimization problem. This nestled structure is typical of bilevel problems and shows why they are inherently hard to solve. Among bilevel programming problems an interesting subclass is formed by problems in which all the functions are linear. These kind of problems, which are the most studied in the literature, are known as Bilevel Linear Problems (BLPs). There are several theoretical results on BLPs in which all the variables are continuous and many solution methods have been developed and successfully applied to real applications. Conversely, the integer case has not been sufficiently investigated and there are only few preliminary results and methods designed to solve special cases.

The main focus of this dissertation are BLPs in which a subset or all the variables are integer. Despite this class of problem can be potentially used to model a very large number of optimization problems with a combinatorial structure, very efficient methods to cope with large size problems have not been developed yet. In this research work we start from some existing results in order to improve the performance of methods proposed in the literature and we also present novel approaches that are remarkable improvements in the state of the art of BLPs. In the first part of this dissertation we introduce the main properties and the existing results on BLPs in general. We present notations, main

definitions, similarities and differences with other existing problems and an overview of applications. Great attention is dedicated to the state of the art of BLPs solution methods in order to point out whether and how these approaches can be extended to the integer case. The central part of the dissertation is composed of two main chapters. In the first, we investigate discrete BLPs in which the leader's variables are integer. This class of problems is also known as Discrete–Continuous Bilevel Linear Problems (DCBLPs). It is known that such problems are equivalent to continuous bilevel linear problems in which the integrality requirements are relaxed and the leader's objective function is modified including a concave penalty function weighted by a parameter. The equivalence holds for a sufficiently large value of the parameter and a valid lower bound for the latter is known in the literature. We provide an improvement of this lower bound and assess the impact of the new lower bound in terms of efficiency on a set of test problems.

Furthermore we propose a valid inequality for a generic DCBLP. The basic idea to compute this valid inequality is to relax the DCBLP, analyze the geometry of its feasible set, that has well known properties, and then obtain information on the bilevel–feasible solutions of the DCBLP. By solving an auxiliary bilevel linear problem it is possible to derive a lower bound on the value of the follower's objective function and then reformulate the original problem taking this bound into account. The possible interpretation of the proposed valid inequality is twofold: it can be considered both like a leader's constraint and a follower's constraint. These two interpretations are discussed and a computational comparison is provided.

In the second main chapter, we investigates discrete BLPs in which all the variables are integer. This class of problems is also known as Discrete Bilevel Linear Problems (DBLPs). We explain why a classical branch and bound method designed for integer programming problems fails to find the optimal solution of a DBLP. We propose two new exact methods and compare them to an existing benchmark algorithm. The first method is a branch and cut approach that is based on a similar idea, properly modified, used to solve DCBLPs. The second method is a cutting plane approach in which it is computed a new valid inequality eliminating an integer solution every time it is proved not to be bilevel–feasible. This valid inequality is a non linear constraint which can be linearized introducing a further follower problem in the original formulation.

Finally two heuristic approaches are proposed which exploit and readapt some geometric properties of the feasible set of a relaxed DBLP in order to find a good solution in a reasonable time.

In the final section we give a flavour of some possible applications of bilevel programming in the field of Operation Management. Two interesting applications of bilevel programming with integer variables are addressed. In the first application we study a specific Grid scheduling problem. A set of independent tasks is submitted to a Grid External Scheduler (ES) and have to be assigned by the ES to a set of Grid computing sites, each one controlled by a Local Scheduler (LS), for their execution. The ES can decide to accept a task and assign it to a LS or to refuse the task: in the first case a penalty cost is paid only if the completion time exceeds the task due date, i.e. there is a positive tardiness,

while in the second case a rejection cost is paid. While the ES looks for executing the submitted tasks over the Grid minimizing the total cost for rejecting or delaying tasks, the goal of each LS is maximizing computational resource usage efficiency. This problem has a clear hierarchical structure in which the two decision makers have different objectives and their decisions mutually affect to each other. The decision of the ES to accept or not a task has an impact on the follower's optimization problem and, vice-versa, the optimal scheduling of the follower defines the tardiness of a task and the consequent decision of the leader to accept or refuse a task. We model this problem by means of bilevel programming with integer variables in the leader's and follower's level. After reformulating the latter as a single level mixed-integer program, we propose a heuristic algorithm to cope with large size instances happening in practice.

The second application is a capacitated facility location problem formulated as a bilevel linear problem with a mixed-integer leader's problem. In the model we propose the leader decides which facilities to open and the capacity to install in each facility in order to minimize the total cost, while the follower controls the assignment of a given set of clients to the open facilities with the goal of maximizing his profit. The basic idea is that the objective of the two decision makers are independent. The key assumption is that the follower (e.g. a private company) is not obliged to satisfy all the clients' demand and the leader (e.g. an Authority) cannot control or apply sanctions on him. The leader can only open more facilities or install more capacity on the open ones in order to guarantee that clients' demand is satisfied beyond a certain threshold, hence it pursues a twofold objective, both economical and social. Conversely, the follower aims at finding the most profitable assignment of clients to facilities. The resulting formulation is a DCBLP. Although there exist in the literature exact methods for this class of bilevel problems, they can not be used to solve large size problems. We propose a branch and cut framework, in order to cope with the bilevel structure of the problem and the integrality of a subset of variables under control of the leader. The algorithm is exact in theory but, for the sake of computation, we introduce suitable stopping criteria and test the algorithm on a set of real life benchmark instances available in the literature.

Acknowledgments

First of all I have to thank my advisor, Professor M. Caramia. During these long years he always gave me useful advices, he guided me and represented a constant point of reference for my study and my research activity.

I am thankful to Professor Y. Fathi because I have known an helpful, interested and inspiring person, beyond any possible imagination. My experience at the NCSU was profitable and remarkable not only for my work, but also from a humane perspective and I have especially to thank him and his generosity.

Many thanks to all people I met during this long experience, both colleagues and friends. Thanks to all people that always supported me, in any circumstance: first of all my family, my friends Chiara and Arianna, she is the one who mostly incited, encouraged and helped me and we both know that, without her, I would have not reached this goal.

Finally, thanks to Giuseppe because during these three (and more) years he was always able to make me laugh, also in the most critical moments, donating me the peacefulness that I needed.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Many modern systems are characterized by the presence of multiple decision makers with different and often conflicting objectives whose decisions, actions and strategies have to coexist within the same environment. The optimization problems arising in such decision systems, may not be formulated and solved by means of classical mathematical programming approaches as they clearly show their flaws and may not provide sufficiently realistic results. The solutions obtained making use of well known optimization techniques with a single decision maker may be an extreme approximation of the real behaviour of the system. For this reason, in the last few decades, there has been an increasing attention toward novel and more sophisticated mathematical programming methods for coping with the presence of multiple decision makers and multiple stakeholders and obtaining more likely representations of real complex systems.

Following different research directions, over the last few years several mathematical methods have been proposed and developed with the goal of overtaking the limits of classical single decision maker models. Among the others, multiobjectives optimization and game theory significantly developed as they represent mathematical approaches in which multiple decision makers and their mutual interactions. Multilevel programming problems are strictly related to the latter and allow to model a particular situation in which the decision makers have different authorities and a hierarchical relationship among them is explicitly defined. The main characteristics of multilevel programming are highlighted by Bialas and Karwan [34]:

- interacting decision makers within the same hierarchical system
- each level takes its decision after the decision of upper levels and taking them into account
- each level optimizes its own objective but it is constrained by the decisions of upper levels and by the reactions of lower levels
- one level can influence the decision of other levels affecting both its objective functions and its feasible set of decisions.

Multilevel programming problems were first introduced by Bracken and McGill [37] [38] in the field of defense problems and defined as mathematical programs that contains an optimization problem in the constraints. The term multilvel and bilevel were first used by Candler and Norton [42]. Basically this kind of problems is characterized by a set of decision makers hierarchically related and the decision of one player is constrained by the decision of the other players, and vice-versa. The first contributions and theoretical properties were formalized by Bialas and Karwan [32], Bard and Falk [23] and Bard [20].

Bilevel programming problems are multilevel mathematical programs in which there are only two decision makers. The restricted number of players makes this class more easy to handle compared to a generic multilevel problem and for this reason most of the contributions that can be found in the literature address the bilevel case.

Bilevel programming problems can be defined as optimization problems in which the feasible set is defined by solving a parameterized problem. In order to explicit the hierarchy between the two players, the first decision maker is called *leader* and the second one is called *follower*. We equivalently refer to the leader as the upper level decision maker and to the follower as the lower level one.

From a mathematical standpoint, a bilevel programming problem is formed by two problems included within a single instance. The optimal value of the variables controlled by the follower (we call them lower level variables) is determined by solving an optimization problem parameterized by the variables under control of the leader (we call them upper level variables). Similarly, the leader's optimal solution is computed knowing the optimal solution of the follower's problem. Hence the general structure is made of two nestled problems, an outer and an inner problem. Bilevel problems are inherently hard to solve since, unlike classical mathematical programming problems, the feasible set is defined by solving the inner optimization problem.

1.1 Notation and general assumption

The general formulation of a bilevel programming problem is the following.

$$\begin{aligned} \min_{x,y} \; & F(x,y) \\ \text{s.t.} \quad & G(x,y) \leq 0 \\ & x \in X \\ & y \in \underset{y}{\text{argmin}} \; f(x,y) \\ & \qquad \text{s.t.} \quad g(x,y) \leq 0 \\ & \qquad\qquad y \in Y \end{aligned}$$

By the formulation it can be clearly noticed the hierarchical structure of the two nestled problems. The objective functions $F(x,y)$ and $f(x,y)$ are respectively called leader's and follower's objective

function. For the sake of clearness within the dissertation we equivalently refer to the leader as the upper level decision maker and to the follower as the lower level decision maker. Other similar terms very often used in the literature are outer problem and inner problem to refer to the leader's and the follower's problem respectively. The variables x are defined on set X and are called leader's variables, while variables y are defined on set Y and are called follower's variables. Finally, $G(x, y) \leq 0$ are the leader's constraints and $g(x, y) \leq 0$ are the follower's constraints.

The set

$$S = \{(x, y) \mid x \in X, y \in Y, G(x, y) \leq 0, g(x, y) \leq 0\}$$

is the *feasible set* of the problem. We refer to solutions $(x, y) \in S$ as *feasible* solutions. The set S is also called semi-feasible set by some authors (see Ben-Ayed [26]). This term clearly expresses the basic difference between bilevel optimization problems (and multilevel problems in general) and single level problems: a solution is not only required to satisfy the set of constraints defining S, but it has also to minimize the follower's objective function for a given vector x. Unfortunately, this term is not used in the most relevant references on bilevel problems and for this reason we do not adopt it in the rest of the dissertation.

For each $\bar{x} \in X$ we define the *follower's feasible set* as

$$\Omega_y(\bar{x}) = \{y \mid y \in Y, g(\bar{x}, y) \leq 0\}$$

and the *reaction set* as

$$R_y(\bar{x}) = \underset{y}{\text{argmin}}\{f(\bar{x}, y) \text{ s.t. } y \in \Omega_y(\bar{x})\}$$

i.e. the set of all the solutions which minimize the follower's objective function. We refer to solutions $(x, y) \in S$ such that $y \in R_y(x)$ as *rational* solutions. Notice that a rational solution may not be a feasible solution for the bilevel problem since it may violate one or more upper level constraints. We define the set

$$IR = \{(x, y) \mid x \in X, G(x, y) \leq 0, y \in R_y(x)\}$$

that is called *inducible region* and represents the set of all the feasible solutions that are both optimal for the follower (i.e rational) and feasible for the leader. Te solutions contained in IR are denoted as *bilevel–feasible* solutions. In other words a bilevel programming problem can be generally formulated as

$$\begin{aligned} \min_{x,y} \ & F(x, y) \\ \text{s.t.} \quad & (x, y) \in IR \end{aligned}$$

It is clear that the difference between IR and S originates from the optimality requirement of the follower's problem.

The main mathematical complexity of bilevel programming stems from the non convexity of the inducible region on which the leader's objective function is computed. Even in the very simple case in which all the functions are linear and the variables continuous, the inducible region is a non convex set. Despite this, some remarkable geometric properties hold which are fundamental to develop efficient solution methods. We provide a description and some examples of them in the next chapter focusing our attention on a special case in which all the functions of the model are linear.

In the rest of the dissertation we assume the feasible set S compact and non empty.

Proposition 1. *If the feasible set S is compact and non empty, there exists at least a $\bar{x} \in X$ such that the follower's feasible set $\Omega_y(\bar{x})$ is non empty, but it may also be unbounded.*

By construction $S \subseteq \{X \times \Omega_y(x)\}$ and since S is non empty, there exists at least a solution $(\bar{x}, \bar{y})$ such that $(\bar{x}, \bar{y}) \in S$ with $\bar{x} \in X$ and $\bar{y} \in \Omega_y(\bar{x})$, that is $\Omega_y(\bar{x}) \neq \emptyset$. Thus, the non emptiness property is preserved from S to $\Omega_y(x)$ for at least a $x \in X$. The same results does not hold for the property of compactness. Due to the lack of constraints $G(x, y)$, the set $\{X \times \Omega_y(x)\}$ may be unbounded below or above and the follower's problem may not be finite. Consider the following case:

$$\Omega_y(x) = \{y \mid y \in \mathbb{R}, -2x + y \leq 1, x + y \leq 4\}$$

for each $x \in \mathbb{R}_+$ the set is non empty and unbounded below. If the leader's constraint $x - 5y \leq -4$ is added, the feasible set S is compact and non empty. If $-\nabla_y f(x, y) \leq 0$ the reaction set $R_y(x)$ is unbounded below and the bilevel problem is unbounded, too, although the feasible set S is a polytope. From this simple example it is clear that the upper level constraints $G(x, y) \leq 0$ play a fundamental role in defining the existence of optimal solutions as pointed out in Vicente et al. [147], Audet et al. [11] and Marcotte and Savard [109]. We investigate these properties in the next chapter for the linear case.

1.2 Stability problem

One of the main features of bilevel programming problems is the possibility that for a given value of x the follower's problem is not well posed and, without further assumptions, the bilevel formulation is meaningless. A generic bilevel–feasible solution is required to be the best response for the follower according to the leader's solution. The reaction set $R_y(x)$ is in general a multi-valued mapping of the leader's variable x. For a given vector $\bar{x}$ it may happen that $R_y(\bar{x})$ is not uniquely defined and there are two (or more) rational solutions, for instance $(\bar{x}, y_1)$ and $(\bar{x}, y_2)$. Assuming that both satisfy the upper level constraints, the follower is indifferent between them, but they may yield different values for the leader's objective function, that is $F(\bar{x}, y_1) \neq F(\bar{x}, y_2)$. In such case the solutions are called *non stable*. This happens every time there are multiple optimal solutions in the follower's feasible set

$\Omega_y(x)$ for a given x, i.e. $R_y(x)$ is not a singleton.

In the literature two different modelling approaches are studied based on a different initial assumption. The first approach is called *optimistic* or *weak* approach. The general assumption is that, if the reaction set is not a singleton, the leader can choose the solution that suits him best. More formally, the optimistic or weak problem is:

$$\begin{aligned} \min_{x,y} \; & F(x,y) \\ \text{s.t.} \quad & G(x,y) \leq 0 \\ & x \in X \\ & y \in R_y(x) \end{aligned}$$

Notice that the latter is equivalent to the general formulation we presented in the previous section. The second approach is called *pessimistic* or *strong* approach. In this case the leader can not choose among the follower's multiple rational solutions. For this reason the worst follower's reaction from the leader's perspective is taken into account. The corresponding formulation is:

$$\begin{aligned} \min_{x} \; & F(x,y) \\ \text{s.t.} \quad & G(x,y) \leq 0 \\ & x \in X \\ & y \in \underset{y \in R_y(x)}{\text{argmax}} \; F(x,y) \end{aligned}$$

This two approaches are deeply different from a mathematical point of view, but also in terms of practical applications. Let us consider this simple linear bilevel problem (the optimistic formulation is used):

$$\begin{aligned} \min_{x,y} \; & -x - 100y_1 - y_2 \\ \text{s.t.} \quad & 0 \leq x \leq 1 \\ & y \in \underset{y}{\text{argmin}} \; -y_1 - y_2 \\ & \quad \text{s.t.} \quad y_1 + y_2 = x \\ & \qquad\quad y_1, y_2 \geq 0 \end{aligned}$$

When $x = 1$, the reaction set $R_y(x) = \{y_1, y_2 \geq 0 \mid y_1 + y_2 = 1\}$ which is not a singleton. Any bilevel–feasible solution for $x = 1$ is non stable since the optimal value for the leader significantly changes according to the best response of the follower. The optimistic solution is $(1, 1, 0)$ with $F(1, 1, 0) = -101$ and the pessimistic solution is $(1, 0, 1)$ with $F(1, 0, 1) = -2$. Solving this problem without a general assumption on the approach used to cope with instability, implies that the problem is not well posed. Moreover, that may happen that the optimal solution of a bilevel problem exists only under the optimistic approach.

Theorem 1. *(Dempe [54]) Consider a bilevel programming problem with positive variables, $x \geq 0$ and $y \geq 0$. Let the feasible set S be non empty and compact, if there exists a solution (x, y) such that $G(x, y) \leq 0$ with $y \in R_y(x)$, $x \geq 0$ then the optimistic formulation admits at least one optimal solution. This is not true for the pessimistic formulation.*

The theorem states that, under the same conditions, the existence of an optimal solution is not guaranteed for both the approaches, but only for the optimistic one. In Bard and Falk [23] this phenomenon is illustrated with the following example (once again the optimistic formulation is used):

$$\begin{aligned}
\min_{x,y}\ & (2y_1 + 3y_2)x_1 + (4y_1 + y_2)x_2 \\
\text{s.t.}\ & x_1 + x_2 = 1 \\
& x_1, x_2 \geq 0 \\
& y \in \operatorname*{argmin}_{y} -(x_1 + 3x_2)y_1 - (4x_1 + 2x_2)y_2 \\
& \quad \text{s.t.}\ \ y_1 + y_2 = 1 \\
& \qquad\ \ y_1, y_2 \geq 0
\end{aligned}$$

The reaction set is easy to compute:

$$R_y(x) = \begin{cases} (1,0) & \text{if} \quad x_1 < \frac{1}{4} \\ y_1 + y_2 = 1 & \text{if} \quad x_1 = \frac{1}{4} \\ (0,1) & \text{if} \quad x_1 > \frac{1}{4} \end{cases}$$

Considering the upper level constraints, the leader's objective function can be restated as:

$$F(x_1) = \begin{cases} -2x_1 + 4 & \text{if} \quad x_1 < \frac{1}{4} \\ 2y_1 + \frac{3}{2} & \text{if} \quad x_1 = \frac{1}{4} \\ 2x_1 + 1 & \text{if} \quad x_1 > \frac{1}{4} \end{cases}$$

There is a step discontinuity for $x_1 = \frac{1}{4}$ and the optimal solution depends on follower's reaction (i.e. y_1). In the optimistic approach, given a set of rational solutions, the leader can control both the upper and lower level variables: in this case he sets $y_1 = 0$, the optimal solution occurs at $x_1 = \frac{1}{4}$ and the leader's objective function is $F(x, y) = \frac{3}{2}$. In the pessimistic formulation it is assumed that the worst solution for the leader is selected by the follower, thus $y_1 = 1$. If follows that if $x_1 = \frac{1}{4}$ then $F(x_1) = \frac{7}{2}$, but adding a small $\varepsilon > 0$ to x_1, the leader's objective function is $F(x_1 + \varepsilon) = \frac{3}{2} + 2\varepsilon < F(x_1)$. Since we can arbitrarily reduce ε, the problem does not have a finite optimal solution. In Figure 1.1 the plot of $F(x_1)$ is depicted in both the optimistic (a) and pessimistic approach (b).
In general, the optimistic approach models a form of cooperative relation between the decision makers, because it is assumed that the leader can influence the follower in his reaction. This assumption is

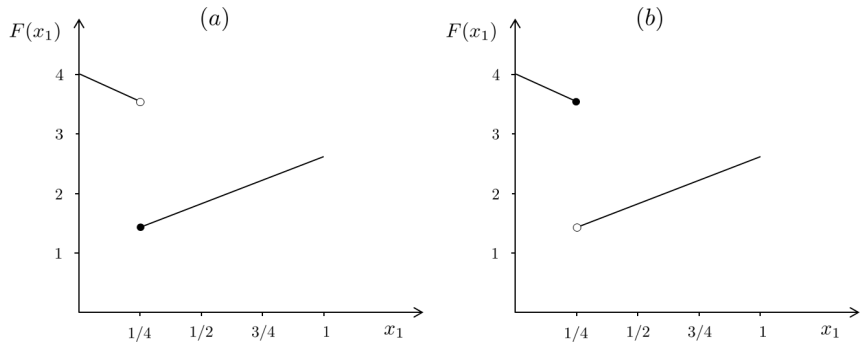


Figure 1.1. Optimistic and pessimistic approach

quite realistic in many applications, for instance when the leader is a public authority that can employ a form of control on follower's behaviour. In the pessimistic approach the basic assumption is that, in a condition of uncertainty and great risk aversion, the leader tries to optimize his objective function taking into account the worst possible reaction of the follower in every circumstances. The first approach is typically used when it can be assumed that the follower can somehow participate to the leader's benefits, while the second approach is mostly used when the leader has a small risk tolerance.

Many attempts have been made to overcame the stability problem. Bialas and Karwan [34] propose to modify the follower's objective function introducing a small contribution of the leader's objective function, that is $f(x,y) = f(x,y) + \varepsilon F(x,y)$ for a suitably small $\varepsilon > 0$. Dempe [54] propose, whenever a non stable solution (x,y) is found, to perturb the leader's variables in order to obtain a reaction set for the follower that is a small neighborhood of y. Different perturbation techniques are proposed and described. In Bianco et al. [35] a stability test is proposed to assess whether an optimal solution is stable or not. Let us assume that $(\bar{x},\bar{y})$ is a bilevel– feasible solution. By solving the following problem:

$$\begin{aligned} \max_{y} \; & F(\bar{x},y) \\ \text{s.t.} \quad & (\bar{x},y) \in S \\ & f(\bar{x},y) \leq f(\bar{x},\bar{y}) \end{aligned}$$

if the solution is non stable a new bilevel–feasible solution is found which is greater than or equal to $F(\bar{x},\bar{y})$. If the difference between the two solutions is greater than a given threshold, the leader can decide to refuse the non stable solution and find a new bilevel-feasible solution that is stable through an heuristic approach. In Gzara [71] a non stable solution is considered infeasible and is discarded by a cutting plane method. Other techniques to tackle the stability problem can be found in Parraga [123], Charnes et al. [48]. The reader is referred to Lignola and Morgan [96] and Loridan and Morgan [99] for a complete discussion on these issues in the case of Stackelberg games.

In the rest of the dissertation we address the *optimistic approach*. All the formulations refer to this mathematical approach and the general assumption behind this is a form of semi-cooperation between the two decision makers as previously described.

Finally note that the well known class of max-min problems has an interesting property regarding the stability issue. In the literature max-min problems have been extensively investigated over the years and there are several applications of these models, see Danskin [52]. Mathematically, max-min problems are bilevel programming problems where $f(x, y) = -F(x, y)$, thus the two decision makers optimize the same objective function but in opposite directions: the follower is assumed to be an external interdictor/attacker who wants to minimize (maximize) the leader's profit (cost). These models are typically used for robust optimization problems or worst case analysis (e.g. for designing and planning protection of the most critical parts of an infrastructure). The main feature of max-min problems, in terms of bilevel programming properties, is that the stability of solutions is always guaranteed. Indeed, in case of multiple optima for the follower, these solutions yield the same objective for the leader and both the decision makers are indifferent among them. In this case it is not necessary to make further assumption as the stability problem never occurs. An interesting class of max-min problems, that is rich of applicative examples, is represented by interdiction problems. The reader is referred to DeNegre [56] for an overview of interdiction problems and their links to bilevel programming.

1.3 Computational complexity

Bilevel programming problems are NP-hard problems since the nestled structure of leader's and follower's problems makes them very hard to be solved. There are many results on computational complexity of bilevel programming and several polynomial reductions to standard combinatorial problems have been proposed. Jeroslow [78] was the first to prove the NP-hardness of bilevel linear programming problems, Vicente et al. [146] proved that checking both strict local and local optimality in bilevel linear programming is NP-hard through a reduction to the problem of 3-SAT. A shorter proof was given by Ben-Ayed and Blair [27] who proved that a binary knapsack problem can always be reformulated as a bilevel linear problem.

These results were strengthened by Hansen et al. [74] who proved that bilevel linear problems are strongly NP-hard. The authors proposed a reduction of a bilevel linear problem to the KERNEL problem. The latter consists of finding a certificate on the existence of a stable subset of nodes K that is also absorbing (i.e. there is an arc (i, j) such that $i \in K$ and $j \notin K, \forall j$). Given a directed graph $G = (N, V)$, let us consider the following max-min problem:

$$\min_x \max_y \sum_{i=1}^{|N|} y_i \tag{1.1}$$

$$\text{s.t.} \quad x_i + x_j \leq 1 \quad \forall (i,j) \in E \tag{1.2}$$

$$x_i \geq 0 \quad \forall i \in N \tag{1.3}$$

$$x_i + y_i \leq 1 \quad \forall i \in N \tag{1.4}$$

$$x_i + y_j \leq 1 \quad \forall (i,j) \in E \tag{1.5}$$

$$y_i \geq 0 \quad \forall i \in N \tag{1.6}$$

Constraints (1.2) guarantee that subset K is stable, while constraints (1.4) and (1.5) are necessary to force the leader to choose an absorbing subset for minimizing his objective function. The variables $x_i = 1$ in the optimal solution correspond to the nodes of the kernel.

Theorem 2. *(Hansen et al. [74]) A graph $G = (N, V)$ has a kernel K if and only if the optimal solution of the max-min problem (1.1)-(1.6) is zero.*

The linear max-min problem (1.1)-(1.6) can be easily reformulated as a bilevel problem with linear constraints and linear objective functions in opposite verse, hence the KERNEL problem can be polynomially reduced to a bilevel problem. The KERNEL problem is strongly NP-hard (Garey and Johnson [67]), thus bilevel programming problems are strongly NP-hard, too.

1.4 Related problems

Although in the last few years bilevel programming has received an increasing interest by the scientific community as a stand alone problem, in the first studies it was strictly associated to multiobjective problems and was tackled using well known techniques of multiobjective optimization. Actually, there are several problems that can be considered related to bilevel programming problems. We present a description of the most remarkable ones in the perspective of this dissertation.

1.4.1 Max-min problems

In general max-min problems can be considered as noncooperative zero sum game with perfect information of both players. As we previously pointed out, a max-min problem can be reformulated as a bilevel problem with objective functions of opposite verse. Falk [60] was the first to show the equivalence between a two stage max-min problem, where the decisions are taken in sequential order, and nonconvex problems. The author investigates the following Linear Max-Min problem (LMM):

$$(LMM) \quad \max_x \min_y c_1^T x + c_2^T y$$
$$\text{s.t.} \quad (x, y) \in S$$

defined on the compact set $S = \{(x, y) \mid x \in \mathbb{R}^n_+, y \in \mathbb{R}^m_+, Ax + By \geq b\}$.

Theorem 3. *(Falk [60]) Problem LMM is equivalent to a nonconvex program with a piecewise linear and convex objective function and defined on the linear polyhedron $S(x)$, that is the projection of S onto the variable x space.*

Theorem 4. *(Falk [60]) Problem LMM always has an optimal solution (x^*, y^*). Moreover there is an optimal solution such that x^* is a vertex of $S(x)$.*

Theorem 5. *(Falk [60]) There is an optimal solution (x^*, y^*) of LMM which is a vertex of S.*

Bard and Falk [23] show that these results can not be extended to a general bilevel problem since they are based on the assumption $f(x, y) = -F(x, y)$, but the same results of Theorems 4 and 5 are proved by Bialas and Karwan [33] for the linear case. In the next chapter the main geometric properties of bilevel linear problems are presented and commented.

The same max-min problem is addressed by Gallo and Ülkücü [65], Konno [89], Alarie et al. [9], Audet et al. [14] and Marcotte and Savard [109] as a possible reformulation of a disjoint bilinear programming problem. Under the assumptions of separability and boundedness of the feasible set, the latter can be reformulated as two nested problems with opposite objective functions defined on the same feasible set.

All the wide class of problems denoted as bottleneck problems, where the objective is to maximize the performance of the most critical element of a system in order to affect the global efficiency, are basically max-min or min-max problems and can be considered like special cases of bilevel programming problems. One of the best known class of bottleneck problems is represented by the centre and median problems on graphs introduced by Hakimi [72] [73]. In the centre problem it is required to choose the location of a facility in order to minimize the maximum distance of clients from it, while in the median problem the objective is the minimization of the total distance. Note that, once a facility is located on a node of the graph, the distance from a client to the facility is computed solving a shortest path problem that represents the follower problem of the bilevel model.

1.4.2 Game theory

Bilevel programming has a natural game theoretic interpretation in terms of Stackelberg games. These games were introduced by Stackelberg [139] as noncooperative dynamic games with perfect and complete information: the players move in sequential order, they know the payoff function and the set of feasible strategies of all the others opponents and in every stage of the game the player who moves exactly knows all the strategies implemented by the previous players. Unlike classical static games, there is a clear order of play that defines which player is the leader (who moves first) and who is the follower (who moves after). Although Stackelberg game is a dynamic game, in the literature it is often defined as static to distinguish the single round game from the repeated one. If the number of players is two, this framework clearly represents a bilevel programming problem. Vicente and Calai [145]

highlight that when the reaction set is not a singleton and stability problem may occur, a solution of the Stackelberg game may not be also a solution of the bilevel problem and the equivalence between the two problems has to be treated carefully.

Stackelberg games have been widely used to model different market scenarios and multilevel economic systems and their main feature is the hierarchical and sequential structure. Bard and Falk [23] highlight the relation between bilevel programs and bimatrix games; indeed, the latter are not dynamic games and the two players are assumed to move simultaneously. The authors show how the result of a bilevel problem instance changes in the bimatrix game and in the two Stackelberg game in which the order of play is inverted. Under a suitable assumption of sequential rationality, Selten [130] introduced the concept of subgame perfect equilibrium. This definition extends the concept of Nash equilibrium (Nash [116]) adding the requirements that the strategy of each player is optimal in every stage of the game and can be viewed as an application of the Bellman's optimality principle of dynamic programming to game theory. Even if computing Nash equilibrium is still a very challenging task and efficient methods in the general case do not exist, algorithm game theory is a developing subfield of game theory and aims at computing equilibrium points in efficient way. The reader is referred to the book of Nisan et al. [118].

Finally Bialas and Chew [33] and Bialas [30] investigate the mathematical properties of coalition games in order to show the effect of coalition formation in $n-$Stackelberg games in terms of overall inefficiencies reduction.

1.4.3 Multiobjective and multicriteria problems

The presence of multiple decision makers is one of the main characteristics of modern systems in which the optimal solution is often the best trade off among different alternatives. In these problems the concept of optimal solution can not be applied in a straightforward manner unlike single decision maker optimization problems. The definition of optimal solution needs to be revised and extended to introduce the notion of efficient solution, as it was defined by Pareto [122]. Here is a general formulation of a multiobjective problem

$$\begin{aligned} &\min_{x} \ [f_1(x), f_2(x), \dots, f_k(x)]^T \\ &\text{s.t.} \quad x \in \mathcal{F} \end{aligned}$$

and a solution x^* is defined Pareto optimal if it does not exist a solution $x \in \mathcal{F}$ such that

$$\begin{aligned} f_i(x^*) &\leq f_i(x) \quad \forall i = 1, \dots, k \\ f_j(x^*) &< f_j(x) \quad \text{for at least a } j \in \{1, \dots, k\} \end{aligned}$$

There have been many attempts to study bilevel programming problems as biobjective problems, especially in the first theoretical studies. Bard [20] noticed that for bilevel linear problems the first

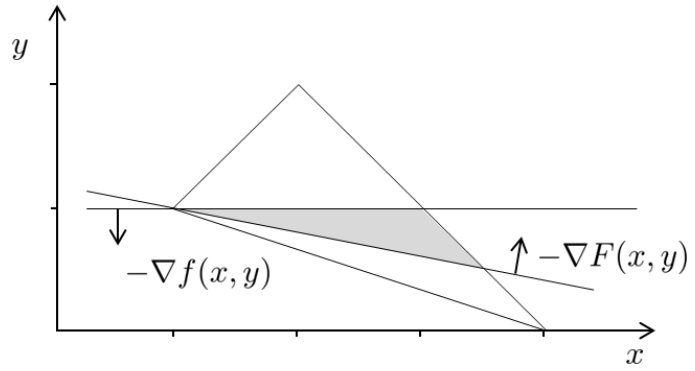


Figure 1.2. Set of solutions dominating point $(1,1)$

order optimality necessary conditions coincide with the optimality conditions of a single level problem defined on the same feasible set and whose objective function is $\alpha F(x,y) + (1-\alpha)f(x,y)$. He proposed to optimally solve a bilevel problem treating it as a single level biobjective problem and iteratively increasing the value of α until the solution computed lies on the reaction set. A similar approach, based on solving the weighted sum of the two objective functions, was proposed by Ünlü [143]. It is a well know result (Geoffrion [69]) that this method, known as scalarization method, is able to compute a Pareto efficient solution. Nevertheless, the optimal solution of a bilevel problem is not necessarily Pareto efficient, thus these methods are not exact. Let us consider the following example:

$$
\begin{aligned}
\min_{x,y}\ & -x-5y \\
\text{s.t.}\quad & x \geq 0 \\
& y \in \operatorname*{argmin}_{y}\ y \\
& \quad \text{s.t.}\quad x-y \geq 0 \\
& \qquad\quad x+y \leq 4 \\
& \qquad\quad x+3y \geq 4 \\
& \qquad\quad y \geq 0
\end{aligned}
$$

The bilevel optimal solution is $(1,1)$ with leader's and follower's objectives equal -6 and 1, respectively. It is simple to note that $(1,1)$ is not a Pareto optimal solution as it is dominated, for instance, by $(3,1)$. In Figure 1.2 the shadowed area represents the set of solutions dominating $(1,1)$.
This result is coherent with the game theoretical interpretation of bilevel problems: in fact, the equilibrium point of a Stackelberg game is not required to be Pareto optimal. In multiobjective programming, unlike bilevel problems, the objective functions are controlled by the same decision maker on a common feasible set. Haurie et al. [75] formally explain why the Bard's result is not always true, but highlight that the solution computed has the interesting properties of being both rational and Pareto efficient. Wen and Hsu [148] tried to define a sufficient condition under which the correspondence between the bilevel and the biobjective problems holds, that is $\nabla_y F(x,y)^T \nabla_y f(x,y) \leq 0$. Marcotte and Savard [107]

showed that this result is false and the only case in which an equivalence between the two problems holds is when $\nabla_y F(x,y) = \lambda \nabla_y f(x,y)$ with $\lambda \geq 0$: it means that there ia a form of cooperation between the leader and the follower and the optimal solution must be Pareto optimal.

The first author who correctly proved the existence of a relationship between bilevel linear and multiobjective problems was Fülöp [64]. The author understood that this link is valid if a multiobjective problem with more than two objectives is taking into account. He addressed the following bilevel linear problem:

$$\begin{aligned}
&\min_{x,y} c_1^T x + c_2^T y && (1.7)\\
&\text{s.t.} \quad x \in \mathbb{R}^n_+ && (1.8)\\
&\qquad y \in \operatorname*{argmin}_y d_1^T x + d_2^T y && (1.9)\\
&\qquad\quad \text{s.t.} \quad Ax + By \geq b && (1.10)\\
&\qquad\qquad y \in \mathbb{R}^m_+ && (1.11)
\end{aligned}$$

with $(n+m)$ variables, $c_1, d_1 \in \mathbb{R}^n$, $c_2, d_2 \in \mathbb{R}^m$, $A \in \mathbb{R}^{q \times n}$, $B \in \mathbb{R}^{q \times m}$ and $b \in \mathbb{R}^q$. Let us consider the matrix

$$A' = \begin{bmatrix} A \\ \mathbb{I} \end{bmatrix}$$

where $\mathbb{I}$ is the identity matrix $n \times n$, thus $A' \in \mathbb{R}^{(q+n)\times n}$. Let $\bar{A}$ be a $r \times n$ submatrix of A' and let $r = rank\, \bar{A}$. Moreover, let the criterion matrix $\mathcal{C}$ be defined as

$$\mathcal{C} = \begin{bmatrix} \bar{A} & O \\ -\mathbb{1}^T \bar{A} & 0_2^T \\ 0_1^T & d_2^T \end{bmatrix}$$

where 0_1^T and 0_2^T are zero row vectors of dimension n and m, respectively, and O is a $r \times m$ zero matrix. The criterion matrix C has dimension $(r+2) \times (n+m)$. It is now possible to define a new multiobjectives problem with $(r+2)$ objective functions as

$$\begin{aligned}
&\min_{x,y} \mathcal{C} \begin{bmatrix} x \\ y \end{bmatrix} && (1.12)\\
&\text{s.t.} \quad Ax + By \geq b && (1.13)\\
&\qquad x \in \mathbb{R}^n_+ && (1.14)\\
&\qquad y \in \mathbb{R}^m_+ && (1.15)
\end{aligned}$$

Theorem 6. *(Fülöp [64]) A solution (x^*, y^*) is bilevel–feasible for problem $(1.7) - (1.11)$ if and only if it is a Pareto optimal solution of the problem $(1.12) - (1.15)$.*

Corollary 1. *(Fülöp [64]) Bilevel problem* $(1.7) - (1.11)$ *is equivalent to minimizing the objective function* $c_1^T x + c_2^T y$ *over the efficient set of the multiobjective problem* $(1.12) - (1.15)$.

Finally the author shows that the relationship between bilevel and multiobjective problems is biunique in the linear case. Hence, it always possible to construct a bilevel linear problem starting from a multiobjective problem such that a solution is feasible for the first if and if it is a Pareto optimal solution for the second and vice-versa. This represents the strongest result for the bilevel linear case and proves that a link between these two class of problems exists, but it is necessary to take into account multiobjective problems with more than two objectives. Fliege and Vicente [61] also investigated the relationship between multiobjective and bilevel problems in the non linear case.

To conclude this section, it is interesting to observe how bilevel programming can be used for multicriteria optimization. These problems are very close to multiobjective ones and in the literature they are often referred indistinctly. In multicriteria problems the feasible set is made of a limited number of alternatives and the objective functions represent the criteria used to consider one alternative better than another. Methods ELECTRE (Roy [125]) are some of the most famous techniques used to tackle these kinds of problems. The rationale behind these methods is to define a domination relationship between each pair of alternatives and then build a graph with a node for each alternative and an edge for each domination relationship. The final step is to detect the kernel of this graph which constitutes the subset of alternatives not dominating each other and dominating all the other alternatives. As we previously pointed out, finding the kernel of a graph can be formulated as a bilevel programming problem.

1.4.4 Stochastic programming problems

Stochastic problems are generally used to model optimization problems under uncertainty conditions. In this kind of problems a subset of variables are assumed to be stochastic. The main challenge of the decision makers is to cope with uncertainty and take a decision now that involves uncertain future scenarios. The most used mathematical approach to model these problems is called two stage stochastic optimization. The main idea is to partition the set of variables into two subsets, a group of variables regarding decisions that must be taken before the random occurrences (first stage) and a group of variables regarding decisions that must be taken in the future once the uncertainty is revealed (second stage). Here is the general implicit formulation of a linear two stage stochastic problem:

$$\begin{aligned} \min_{x,\xi} \; & c^T x + \mathbb{E}_\xi Q_\xi(x, \xi(\omega)) \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

where $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{q \times n}$, $b \in \mathbb{R}^q$. The vector $x \in \mathbb{R}^n_+$ represents the first stage variables and the function $Q_\xi(x, \xi(\omega)$ is defined as

$$Q_\xi(x, \xi(\omega)) = \min_y \{q(\omega)^T y(\omega) \mid y(\omega) \geq 0, W(\omega)y(\omega) = h(\omega) - T(\omega)x\}$$

where $y(\omega) \in \mathbb{R}^m_+$ are the second stage variables, $\omega \in \Omega$ is a random event and $T(\omega) \in \mathbb{R}^{q \times n}$ and $W(\omega) \in \mathbb{R}^{q \times m}$ are known as the technology matrix and the recourse matrix, respectively. As the second stage parameters and variables depend on the outcome of random events ω, we can define a stochastic variable $\xi(\omega)$. Note that the set of constraints on the stochastic variables are defined for each possible outcome ω, hence the feasibility of solution is always guaranteed no matter the random occurrences. The objective function of the model is the sum of a deterministic cost vector and the expected value of a random variable taking into account the future outcomes of random event ω. The explicit formulation of the problem is

$$\begin{aligned} \min_x \; & c^T x + \mathbb{E}_\xi[\min_y q(\omega)^T y(\omega)] \\ \text{s.t.} \quad & Ax = b \\ & T(\omega)x + W(\omega)y(\omega) = h(\omega) \\ & x, y(\omega) \geq 0 \end{aligned}$$

From the explicit formulation the link between bilevel and stochastic problems can be noted in a straightforward manner. In the simplest case the stochastic variables $\xi(\omega)$ is assumed to be discrete and a small number of scenarios is considered. In this case the stochastic model is easy to handle, but in general the nestled structure of the problem makes it hard to be solved as well as bilevel problems. The reader is referred to monographs of Birge and Louveaux [36] and Kall and Wallace [81] for a more detailed overview of stochastic programming.

1.4.5 Mathematical programming with equilibrium constraints

One of the first and most immediate relationships of bilevel programming problems with other problems concerns Mathematical Programming with Equilibrium Constraints (MPECs). Colson et al. [49] show that bilevel problems are a special subclass of MPECs, since the first problem can always be formulated as the second with proper precaution. Bilevel problems can be easily considered as MPECs in which the equilibrium constraints are the optimality conditions of the follower's problem. MPECs can be viewed as bilevel problems with a variational inequality in the upper level problem. Let us consider the general formulation of an MPEC

$$\begin{aligned} \min_{x,y} \; & F(x, y) \\ \text{s.t.} \quad & (x, y) \in S \\ & y \in C(x) \end{aligned}$$

where $S \subseteq \mathbb{R}^{n+m}$ is closed and non empty and $C(x)$ is the set of $y \in \Omega_y(x)$ such that y is the solution of the variational inequality VI (ψ, Ω_y) defined by the function $\psi : \mathbb{R}^{n+m} \to \mathbb{R}^m$ and the closed convex set $\Omega_y(x) \subset \mathbb{R}^m$:

$$(v - y)^T \psi(x, y) \geq 0 \qquad \forall v \in \Omega_y(x)$$

If $\psi(x, y) = \nabla_y f(x, y)$ and $\Omega_y(x) = \{y \mid y \in Y, g(x, y) \leq 0\}$, the variational inequality VI (ψ, Ω_y) defines the stationarity conditions of the minimization problem:

$$\begin{aligned} \min_y \ & f(x, y) \\ \text{s.t.} \quad & y \in \Omega_y(x) \end{aligned}$$

thus the resulting formulation is equivalent to a bilevel problem. The reader is referred to the Luo et al. [103] for a comprehensive investigation of MPECs and their possible applications.

1.5 Applications

There are several examples of applications of bilevel programming models to a wide range of real problems, especially for bilevel linear programs that are the problems we mainly address in this dissertation. This is due to the great capacity of bilevel programming of representing the hierarchical nature of several decision making processes. A general framework that is common to all applications is the presence of a system that is controlled, designed and managed by one decision maker, but it is used by another. Hence the interests of the two decision makers are usually different and often conflicting.

Interesting surveys on applications of bilevel programming to real life problems are proposed, among the others, by Colson et al. [49], Brotcorne [39] and Bard [21]. Bilevel programming has been used to model and solve problems of many different fields. Migdalas [113] highlights the bilevel and hierarchical nature of transportation planning problems: the social interest of the leader, who seeks to maximize the global efficiency of the network, is different from the objectives of the users who choose how to move inside the network with regard to travel time and cost. A similar problem is encountered in highway toll pricing presented by Labbe et al. [94]. Ben-Ayed et al. [28] apply bilevel programming for solving a highway network design problem with real data coming from the Tunisian highways system. Further applications in the transportation field can be found in Marcotte [106] and Gao et al. [66]. The same framework with two decision makers, one managing a system and the other using it, is proposed by Hobbs and Nelson [76] in the energy sector.

Another problem that is addressed in terms of bilevel optimizations is the hazardous material transportation. In this kind of problem, that is still a network design problem, the aim of the leader is to minimize the risk coming from the circulation of hazardous materials on a network. The leader can prohibit the circulation on a link of the network, as in Gzara [71], or can impose a restriction on the amount of traffic over the link, as in Bianco et al [35]. Other examples can be found in Kara and Verter [83].

In Bard [21] two interesting applications of linear bilevel problems can be found in the field of production planning on different levels. The first addresses a production planning problem in which a manufacturer (the ledaer) has to cope with a stochastic demand that varies according to the level of advertising investment. In the second application the leader is the French government that seeks to define the optimal amount of incentives to encourage farmers to produce nonfood crops used for biofuel production. An improved version of the latter model is presented in Bard [25]. In the management field Cassidy et al. [46] propose a bilevel model for the problem of allocating resources from a central government to regional ones for financing local projects. Other examples can be found in Anandalingam [2] and Bard [18].

As we previously mentioned, there is a wide literature on interdiction problems to solve worst case problems and robustness analysis. Scaparra and Church [128] and [129] address the problem of ensuring the best protection of a system with critical infrastructures against possible attacks and disruptions. The model is a bilevel formulation in which the leader decides which facility fortify in order to reduce the impact of disruptive events. Other applications of bilevel programming used to model interdiction problems are in Aksen et al. [7] and [6] and Liberatore et al. [95], while Arroyo and Galiana [10] propose a similar bilevel approach for the terroristic threat problem. Economic models based on the Stackelberg equilibrium problem exploit the theoretical results of bilevel programming. In Smith et al. [137] the authors address the problem of defining the best price strategy for the introduction of a new product on a market in a scenario with a predator competitor. A typical Stackelberg game in an oligopolistic situations is also investigated by Sherali et al. [131] and Kochetov et al. [88].

There is an arising area of research that exploits the theoretical properties of bilevel programming in order to solve other hard mathematical problems, especially in the combinatorial area. In this case bilevel programming problems are used as a subroutine and their efficient solutions may positively impact on other problems. In Lodi and Ralphs [98] the problem of finding the maximally violated valid inequality for a given convex hull is naturally formulated as a bilevel problem. A similar separation problem is addressed by Mattia [111] for the network loading problem. Finally Sahin et al. [126] present a survey on the applications of bilevel programming in civil engineering (traffic management) and chemical engineering (process synthesis).

1.6 General framework and our contributions

In this dissertation our main focus is addressed to bilevel linear problems with a subset or all the variables discrete. As we mentioned, there is a wide literature on the possible applications of bilevel programming to real life problems and in the last few decades a great effort has been made to investigate properties and develop efficient solution methods. Notwithstanding, the research area of discrete bilevel problems is still poorly studied and there is a lack of efficient methods for general instances of such problems. We believe that discrete bilevel programming has not been exploited in all its potentiality,

especially for modelling integer and combinatorial problems, despite it represents a natural and immediate framework for hierarchical and multi decision makers problems. This is mainly due to the computational difficulties of integer bilevel problems, even in the linear case.

In the remainder of the dissertation we investigate potentiality and criticality of this promising area of research as follows. In Chapter 2 we present the most remarkable results of bilevel linear problems. A series of theoretical results are introduce along with an overview of the best performing solution methods existing in the literature. We point out how the problem changes if a set or all the variables are discrete and we introduce a classification of discrete bilevel liner problems. In Chapter 3 we investigate the case in which the upper level variables are discrete and the lower level variables are continuous. An improvement of an existing reformulation technique is presented. Furthermore, we propose a new valid inequality for this class of problems. In Chapter 4 we address the case in which all variables are discrete. We highlight the limits of classical methods for integer problems and propose two new exact methods and two heuristics based on some geometrical properties of bilevel problems. In Chapter 5 we present two applications of discrete bilevel linear problems to a Grid scheduling problem and a facility location problem.

Summing up, these are the main contributions we provide in this dissertation:

- an improvement of an existing reformulation method for bilevel problem with discrete leader's and continuous follower's variables;
- a new valid inequality for the same class of problem
- two new exact methods for bilevel problem with all integer variables
- two new heuristic methods for the same class of problem
- two new applications of discrete bilevel linear programming in the field of industrial engineering.

Finally, in Chapter 6 we provide our final remarks, open research points and possible directions for future research.

Chapter 2

Bilevel Linear Programming

In this chapter we address a particular class of bilevel programming problems that is one of the most studied in the literature. We assume that all the variables are continuous and all the functions are linear, both the constraints and the objective functions. Let us define these problems as Bilevel Linear Problems (BLP). For BLPs many properties and solution methods have been proposed and in the rest of this chapter we provide a description of the state of the art of the best performing algorithms.

The general formulation of the problem we address is the following:

$$\begin{aligned}
\min_{x,y}\ & F(x,y) = c_1^T x + c_2^T y \\
\text{s.t.}\ & Cx + Dy \leq e \\
& x \in \mathbb{R}^n_+ \\
& y \in \operatorname*{argmin}_y\ f(x,y) = d_1^T x + d_2^T y \\
& \qquad \text{s.t.}\ \ Ax + By \leq b \\
& \qquad\qquad y \in \mathbb{R}^m_+
\end{aligned}$$

where $c_1, d_1 \in \mathbb{R}^n$ are the leader's cost vectors in the upper level and lower level objective functions, respectively, $c_2, d_2 \in \mathbb{R}^m$ are the follower's cost vectors in the upper and lower level objective functions, respectively, $A \in \mathbb{R}^{q\times n}$, $B \in \mathbb{R}^{q\times m}$ and $b \in \mathbb{R}^q$ define the set of lower level constraints and $C \in \mathbb{R}^{p\times n}$, $D \in \mathbb{R}^{p\times m}$ and $e \in \mathbb{R}^p$ define the set of upper level constraints. The size of a generic BLP is defined by parameters n, m, p and q. Notice that the follower's problem is parameterized for each value of the leader's variables x. Thus, according to a given $\bar{x} \in \mathbb{R}^n_+$, both the follower's feasible set $\Omega_y(\bar{x})$ and the follower's objective function $f(\bar{x}, y)$ change. Let us write the follower's problem as

$$\begin{aligned}
\min_y\ & f(\bar{x},y) = d_1^T \bar{x} + d_2^T y \\
\text{s.t.}\ & By \leq b - A\bar{x} \\
& y \in \mathbb{R}^m_+
\end{aligned}$$

As a simple matter of fact, once $\bar{x}$ is set by the leader and $\Omega_y(\bar{x})$ is defined, the follower's best solution does not depend on variable x as $d_1^T \bar{x}$ is only a constant and does not impact on the follower's reaction.

Thus, for ease of presentation and for saving notation, in the rest of this dissertation we omit the cost vector d_1 and we assume, without loss of generality, that the follower's objective function does not depend on the x variables, i.e. $f(x, y)$ is replaced with $f(y)$. Recall that the follower problem represents a constraint in the leader's view point, thus roughly speaking we are interested in the follower's best solution and not in the numerical value of its objective function.

2.1 Polyhedral properties

In the introduction we provided the general definitions of feasible set S, reaction set $\Omega_y(x)$ and inducible region IR. Now we characterize them in the linear case. In the following some remarkable results about the geometrical properties of the solutions space of BLPs are reported and commented.

Theorem 7. *(Bard [20]) The inducible region can be written equivalently as a piecewise linear inequality constraint comprised of supporting hyperplanes of S.*

This result implies that, even if the feasible set S is compact and non empty, the inducible region IR is in general a non convex set. Recall that a bilevel problem is equivalent to minimize the leader's objective function over its inducible region. Hence, even in the linear case, BLPs are optimization problems defined over non convex feasible regions and this explains their computational complexity. Notwithstanding, it is still possible to define a weak convexity property for IR.

Theorem 8. *(Bialas and Karwan [33]) If a solution $(x, y) \in IR$ can be written as convex combination of k solutions $(x_i, y_i) \in S$, then $(x_i, y_i) \in IR \quad \forall i = 1, \ldots, k$.*

The latter implies the following two results.

Corollary 2. *(Bialas and Karwan [33]) An extreme point of IR is also an extreme point of S.*

Note that, due to the linearity of $F(x, y)$, in theory a BLP may be reformulated replacing IR with its convex combination and defining a new convex set that is contained in S. Unfortunately this reformulation can not be realized in practice as it requires an explicit description of IR and all its vertexes. Let us consider the following example for illustrating all the previous results.

$$
\begin{aligned}
&\min_{x,y} \; x - 5y \\
&\text{s.t.} \quad x \geq 1 \\
&\qquad y \in \operatorname*{argmin}_{y} \; y \\
&\qquad\quad \text{s.t.} \quad -x + 3y \leq 6 \\
&\qquad\qquad\quad x + y \leq 6 \\
&\qquad\qquad\quad x + 2y \geq 4 \\
&\qquad\qquad\quad x - 2y \leq 3 \\
&\qquad\qquad\quad y \geq 0
\end{aligned}
$$

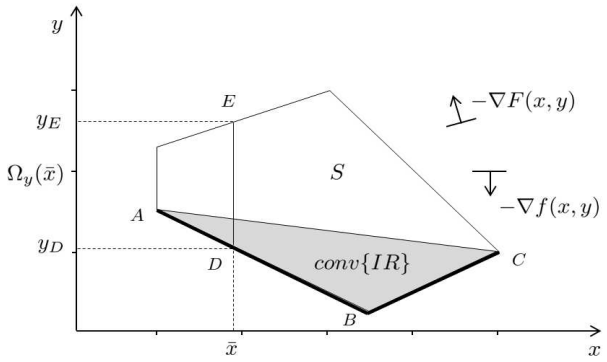


Figure 2.1. Feasible set, reaction set and inducible region

In Figure 2.1 it is easy to observe a graphic representation of the regions associated to the example. The feasible set S is the polytope defined by the white and the shadowed areas. The projection of S onto the leader's space is the set $[1, 5]$. For a given $\bar{x} \in [1, 5]$, the set of follower's solutions contained between points y_D and y_E forms the follower's feasible set $\Omega_y(\bar{x})$. The point y_D is the follower's reaction set $R_y(\bar{x})$ which is a singleton in this case, thus point D is a rational solution. The inducible region IR, represented by the bold lines, is only a small portion of feasible set S and it is fully defined by two supporting hyperplanes of it. The shadowed area is the convex hull of vertexes A, B and C of IR: as expected the latter is contained in S and all the vertexes of IR are vertexes of S according to Corollary 2. The optimal solution occurs at vertex A which is both a vertex of S and IR.

2.1.1 Upper level constraints

One of the main characteristics of bilevel programming problems is the presence of two different set of constraints, the upper level ones and the lower level ones, which impact differently on the definition of the solutions space. In this section we highlight some theoretical results concerning the role of upper level constraints as in the applications we describe in Chapter 5 the presence of these constraints is shown to be not negligible. Moreover, in our opinion, in the literature the effect of the upper level constraints does not seem to be sufficiently investigated and a large number of resolution methods, more or less implicitly, assume they are not present in the formulation.

According to the previous definitions, it is a matter of fact that whenever there are no upper level constraints, that is $p = 0$, a rational solution is also bilevel-feasible. The same result can be extended to the case in which the upper level constraints do not involve the follower's variables: in the previous example $x \geq 1$ is an upper level constraint in which there are no follower's variables and, indeed, every rational solution is also bilevel-feasible. In the same example, if the constraint $x + 2y \geq 4$ is moved to the upper level, the inducible region deeply changes, as it can be observed in Figure 2.2.
The projection of S onto the x space is the interval $[1, 5]$ and $\forall x \in [1, 5]$ the follower's feasible set

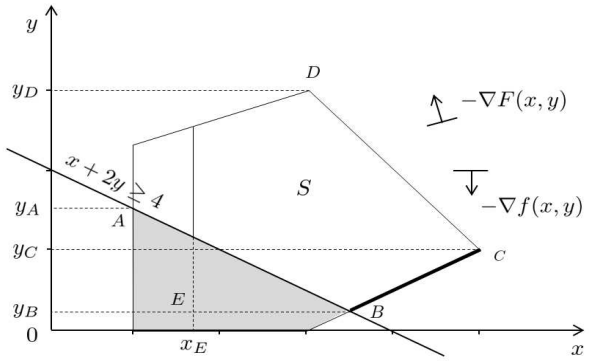


Figure 2.2. Role of the upper level constraints

$\Omega_y(x) \subseteq [0, y_D]$ and the reaction set $R_y(x) \subseteq [0, y_C]$. The interval $[y_B, y_A]$ represents the projection of the upper level constraint $2y \geq 4 - x$ onto the y space: all the rational solutions out of this interval are not bilevel-feasible. Hence, all the rational solutions in $[0, y_B)$ are not feasible in the leader's perspective. Let us consider point E, with $y_E = 0$. Point E is a rational solution, as $y_E \in R_y(x_E)$, but it is not bilevel-feasible. The bold segment $B - C$ is the new inducible region that is still piecewise linear and is a supporting hyperplane of S. The optimal solution is vertex C.

From a geometrical point of view, the difference between the latter two examples occurs because in the second there is an upper level constraint whose projection onto the y space intersects the follower's feasible set for a given x, and cut off some rational solutions. The shadowed area in Figure 2.2 represents the subset of solutions not satisfying the leader's requirements but feasible for the follower. Notice that, if in the example the upper level constraint is replaced with $x + 2y \geq 7$, the only bilevel–feasible solution is vertex C, while for higher value of the right-hand-side the BLP is an empty problem despite S is compact and non empty. This results can be generalized as follows.

Theorem 9. *Given a bilevel linear problem P' with $p \neq 0$, if at least one leader's constraint is moved to the follower problem, the new bilevel linear problem P'' is a relaxation of P'.*

Proof. Let us assume that, for a given x' such that $(x', y) \in S$, the solution (x', y') is rational. Let us consider the upper level constraint $c_j^T x + d_j^T y \leq e_j$. If (x', y') satisfies this constraint, moving $c_j^T x + d_j^T y \leq e_j$ to the follower's problem does not change the rationality of the solution. Conversely, if the upper level constraint is violated, it follows that solution (x', y') is not bilevel–feasible for the original problem. If constraint $c_j^T x + d_j^T y \leq d_j$ is moved to the follower's problem, (x', y') is not longer a rational solution for problem P' since y' does not belong to the new set $\Omega_y(x')$. Let (x', y'') be the new rational solution (note that it must exist since we chose an x' such that $(x', y) \in S$). Two cases may happen: if (x', y'') violates another upper level constraint it is not bilevel-feasible for P'', otherwise P'' admits a bilevel-feasible solution at x' unlike P'. The same rationale can be repeated if the follower does not admit a finite optimal solution, i.e. $R_y(x')$ is not a finite set. This completes the proof. □

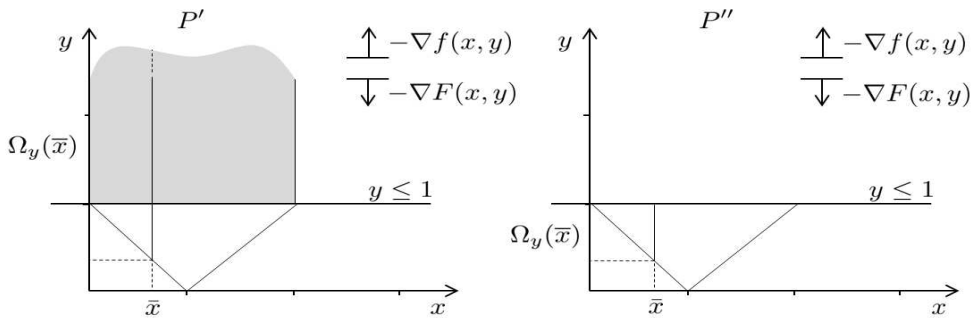


Figure 2.3. Upper level constraints may induce infeasibility

In the proof of Theorem 9 it is interesting to note that, in some particular cases, the upper level constraints play a fundamental role in defining feasibility of BLP. Let us consider this simple example:

$$
\begin{aligned}
(P') \quad & \min_{x,y} y \\
& \text{s.t.} \quad y \leq 1 \\
& \qquad x \geq 0 \\
& \qquad y \in \operatorname*{argmin}_{y} -y \\
& \qquad\quad \text{s.t.} \quad x + y \geq 1 \\
& \qquad\qquad x - y \leq 1 \\
& \qquad\qquad y \geq 0
\end{aligned}
\qquad\qquad
\begin{aligned}
(P'') \quad & \min_{x,y} y \\
& \text{s.t.} \quad x \geq 0 \\
& \qquad y \in \operatorname*{argmin}_{y} -y \\
& \qquad\quad \text{s.t.} \quad y \leq 1 \\
& \qquad\qquad x + y \geq 1 \\
& \qquad\qquad x - y \leq 1 \\
& \qquad\qquad y \geq 0
\end{aligned}
$$

In Figure 2.3 we can see exactly what is stated in the proof of Theorem 9: at $x = \bar{x}$ in P' the follower's feasible set $\Omega_y(\bar{x})$ is unbounded and $R_y(\bar{x})$ is not a finite set, while in P'' the set $\Omega_y(\bar{x})$ is bounded and a bilevel-feasible solution exists. While problem P' is infeasible, problem P'' has a set of multiple optimal solutions $(x, 1)$ with $x \in [0, 2]$.

Shi et al. [135][132][133][134] propose a new definition of bilevel linear problems and state that the formulation they introduce is able to solve instances that the classical model fails to solve. The model they propose is obtained placing the upper level constraints in the follower's problem, thus for Theorem 9 they relax the original formulation of the problem. The authors start from the wrong observation that if S is non empty and compact and $\Omega_y(x) \neq \emptyset \ \forall x$ such that $(x, y) \in S$, a Pareto optimal solution must exist. In fact, as we showed in the previous chapter, a bilevel–feasible solution is not necessarily a Pareto optimal solution and the two properties are distinct concepts. Finally, as observed by Audet et al. [11] and Mersha and Dempe [112], the authors propose a weaker formulation of the problem because they ignore the different role played by the upper and lower level constraints and solve a relaxation of the original BLP.

Finally, in order to understand the meaning of upper level constraints in terms of application on real problems, let us consider the following hazardous material transportation problem. Given a graph

$G = (V, E)$ that represents the road network of a geographical area, the leader wants to minimize the risk associated to the hazmat transportation, while the follower has the objective of minimizing the transportation costs for carrying hazmat materials from an origin s to a destination t. The leader can allow or forbid the transit on a given arc (i, j) and the follower, once the network has been designed by the leader, chooses the shortest path for realizing the transportation. The bilevel model that formulates this problem is:

$$
\begin{aligned}
&\min_{x,y} \sum_{(i,j)\in E} \rho_{ij}\, y_{ij} \\
&\text{s.t.} \quad x_{ij} \in \{0,1\} \\
&\qquad y \in \operatorname*{argmin}_{y} \sum_{(i,j)\in E} c_{ij}\, y_{ij} \\
&\qquad\quad \text{s.t.} \sum_{j|(i,j)\in E} y_{ij} - \sum_{j|(j,i)\in E} y_{ji} = \begin{cases} 1 & i = s \\ 0 & i \neq s,t \\ -1 & i = t \end{cases} \\
&\qquad\quad y_{ij} \leq M \cdot x_{ij} \\
&\qquad\quad y_{ij} \in \{0,1\}
\end{aligned}
$$

Variable x_{ij} is equal to 1 if the leader allows the follower to transport hazmat materials on arc (i, j) and 0 otherwise. Variable y_{ij} is equal to 1 if the follower uses the arc (i, j) in the $s - t$ path and 0 otherwise. A risk ρ_{ij} and a cost c_{ij} are associated to every arc (i, j): the cost is associated to the arc's length, while the risk is due to the presence of critical infrastructures that may be damaged in case of accident. Let us think to a road passing in the city center of a small city, it may have a small transportation cost, but a high risk because of impact and damages that a possible accident may cause. The objective of the follower is to find the least expensive path, while the objective of the leader is to minimize the total risk. The follower problem is a shortest path problem in which constraints $y_{ij} \leq M \cdot x_{ij}$ are added: if the transit on an arc (i, j) is forbidden, i.e. $x_{ij} = 0$, the follower can not choose arc (i, j) in his shortest path problem.

Let us consider the instance in Figure 2.4 with $s = 1$ and $t = 4$.

The optimal solution for the leader has $x^*_{14} = 0$ as this is the only way to forbid follower to transit on arc $(1, 4)$ that represents both the shortest path and the path with the highest risk. The follower's shortest path is $1 - 3 - 4$ of cost 22 and risk 3 and this is the optimal solution of the model. If we move the set of constraints $y_{ij} \leq M \cdot x_{ij}$ from the lower to the upper level, the problem does not apparently change very much. Actually, regardless the choice of the leader, the follower will always select path $1 - 4$ of cost 1 and risk 100. Thus, if the leader sets $x_{14} = 0$, the follower's best response is not bilevel-feasible because of the upper level constraints $y_{ij} \leq M \cdot x_{ij}$. It follows that all bilevel-feasible solutions allow transit on arc $(1, 4)$ and the optimal solution of the problem has risk equal to 100. This result shows that, just modifying the position of a set of constraints, their role in the formulation changes and the solutions space may be completely different.

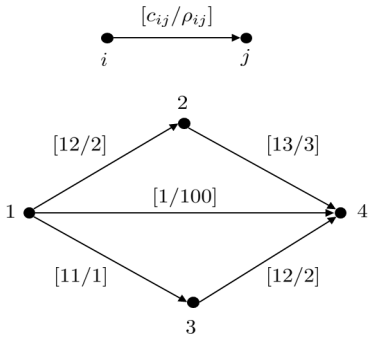


Figure 2.4. A simple instance for a bilevel hazmat transportation problem

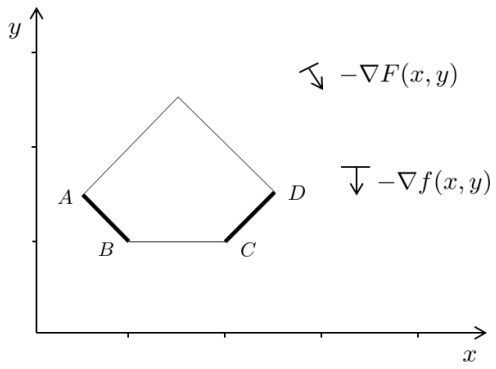
2.2 BLP relaxations

2.2.1 Relaxation via removal of constraints

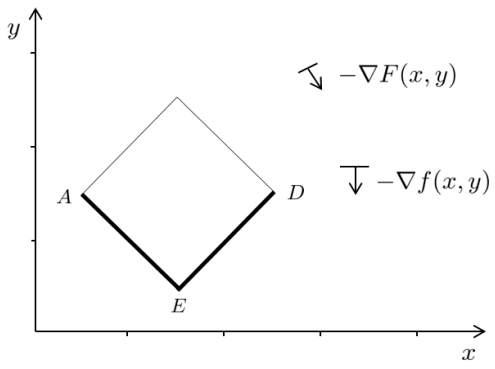
The different impact of upper level and lower level constraints on the bilevel formulation has to be taken into account when we want to relax a BLP by dropping a subset of constraints. Unlike classical mathematical programming models, the new formulation obtained removing one or more constraints does not necessarily provide a valid relaxation. Indeed, this depends on the choice of the constraints we remove. Let us consider the following problem:

$$
\begin{aligned}
\min_{x,y} \; & -x + 3y \\
\text{s.t.} \quad & y \geq 1 \\
& y \in \underset{y}{\text{argmin}} \; y \\
& \quad \text{s.t.} \quad x + y \leq 4 \\
& \qquad\quad x + y \geq 2 \\
& \qquad\quad x - y \geq -1 \\
& \qquad\quad x - y \leq 1
\end{aligned}
$$

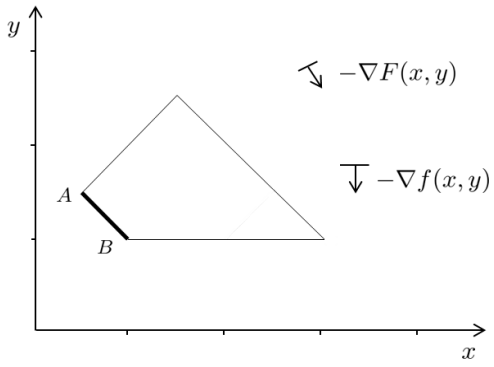
The optimal solution of the BLP is $(2, 1)$, which is vertex C of S, and the optimal value is 1, see Figure 2.5(a). If the upper level constraints $y \geq 1$ is dropped, as depicted in Figure 2.5(b), the inducible region is wider, and the new optimal solution is vertex E, $(1.5, 0.5)$, and the objective function's value is 0. Finally, if the follower's constraint $x - y \leq 1$ is removed, the follower's feasible set is larger, but the new rational solutions of the follower violate the upper level constraint. The inducible region is reduced to segment A–B and the optimal solution is vertex B, $(1, 1)$, which gains an objective function equals 2. Hence, in the second reformulation represented in Figure 2.5(c), dropping a follower's constraint we do not obtain a valid relaxation and the optimal solution is the worst out of the three cases.



(a) Inducible region of the original problem



(b) Removal of an upper level constraint



(c) Removal of a lower level constraint

Figure 2.5. Different effects of removal of constraints

This result, which is apparently unclear, can be easily understood if we think to the inherent meaning of bilevel programming problems. The general framework of bilevel programming is the presence of two non cooperating decision makers. Dropping a lower level constraint means that the set of possible choices for the follower is increased and thus he acquires more contractual power towards the leader. This implies that the solutions space of the leader may reduce and the value of the optimal solution may get worse. On the contrary, if an upper level constraint is removed, the solutions space of the follower does not change, but a larger set of rational solutions may be considered acceptable by the leader. Thus the value of the optimal solution can not be worse and this represent a valid relaxation.

The following proposition sums up the above mentioned results.

Proposition 2. *A BLP can be correctly relaxed dropping a subset of constraints, if and only if these are constraints under control of the leader.*

2.2.2 Single level relaxation

One of the most used and immediate relaxation of a BLP is the so called single level relaxation. It consists of dropping the follower's objective function, turning the bilevel structure into a single level one and assuming that only one decision maker is involved; from a mathematical point of view the leader's feasible set is fully defined by a set of constraints and it is not necessary to solve an inner problem to check feasibility of a solution. The feasible set S does not change and the problem consists of solving the leader's objective function on S.

It is immediate to understand that the single level version of a BLP is a valid relaxation, as a solution that is bilevel–feasible for the BLP is also feasible for the single level problem, but the contrary may not be true.

There is a special case in which the optimal solution of a BLP and of the optimal solution of its single level version coincides. For the sake of simplicity, let us assume that there are no upper level constraints. If the following holds

$$\nabla_y F(x, y) = \nabla f(y)$$

it means that the leader's and the follower's objective function have the same verse, thus minimizing (or maximizing) $F(x, y)$, $f(y)$ is minimized (or maximized) as well. In this case, the optimal solution found solving the single level formulation is also optimal for the BLP.

This result may not hold if we introduce an upper level constraint which depends on the upper level variables x. We now prove the following two theorems, which are necessary and sufficient conditions under which the optimal solution of the single level relaxation of a BLP is also optimal for the original problem. According to these conditions, it is easy to know whether the optimal solution of the relaxation is bilevel–feasible, without solving the inner problem. We assume that S is compact and non empty.

Theorem 10. *If the optimal solution $(\bar{x}, \bar{y})$ of the single level relaxation of a BLP is optimal for the original BLP, then $\nabla_y F(x, y) = \nabla f(y)$ and among the active constraints at $(\bar{x}, \bar{y})$ there is at least one constraint under control of the follower.*

Proof. Let us assume that $(\bar{x}, \bar{y})$ is a vertex of S in which the only active constraints are under control of the leader. Let us define $S' = S \setminus \{Cx + Dy \leq e\}$. From the previous sections we know that $S \subseteq S'$ and that solving the single level relaxation of BLP on S' two cases may occur: a) we found a solution (x', y') on the boundary of S', b) the single level relaxation does not admit a finite solution, i.e. S' is unbounded. In case a), if $(x', y') \equiv (\bar{x}, \bar{y})$, it means that there is at least an active constraint at $(\bar{x}, \bar{y})$ which is not under control of the leader and this contradicts our initial assumption. It follows that $(\bar{x}, \bar{y})$ is an internal point of S', thus there exists a rational solution $(\bar{x}, y^*)$ such that $f(y^*) < f(\bar{y})$. In case b), once again, $(\bar{x}, \bar{y})$ is an internal point of S' and $\bar{y}$ is not a rational solution at $\bar{x}$. In both cases $(\bar{x}, \bar{y})$ is not bilevel–feasible, hence the proof. □

Theorem 11. *Given the optimal solution $(\bar{x}, \bar{y})$ of the single level relaxation of a BLP, if $\nabla_y F(x, y) = \nabla f(y)$ and among the active constraints at $(\bar{x}, \bar{y})$ there are no upper level constraints, $(\bar{x}, \bar{y})$ is also the optimal solution of BLP.*

Proof. If there are no upper level constraints active at $(\bar{x}, \bar{y})$, it means that if we solve the single level relaxation on S', defined as above, the optimal solution found is always $(\bar{x}, \bar{y})$. Thus $(\bar{x}, \bar{y})$ is a rational solution and satisfies the upper level constraints, hence it is bilevel–feasible and optimal for the BLP, which completes the proof. □

In Figures 2.6(a)-(d) there is a graphic explanation of Theorems 10 and 11. In each figure we reported both the feasible set S and some additional upper level constraints.
In Figure 2.6(a) an upper level constraint intersects the inducible region and solution A is a vertex of S in which a lower level constraint is active; A is bilevel–feasible and is the optimal solution of the BLP. In Figure 2.6(b), although a lower level constraint is active at A, unlike the previous case the solution is not bilevel–feasible and the optimal solution of the BLP is vertex B: this shows that the condition stated in Theorem 10 is only necessary and not sufficient. In Figure 2.6(c) an upper level constraint intersects the inducible region but this does not change the optimal solution of the single level relaxation problem and Theorem 11 holds. Finally, in Figure 2.6(d) all the constraints active at vertex A are upper level constraints and by Theorem 11 the solution is not optimal for the BLP.

2.3 Solution methods

Bilevel Linear Problems have been widely investigated over the last two decades and there is a consistent number of contributions in terms of solution methods. Many interesting surveys can be found in the literature, for instance Ben-Ayed [26], Wen and Hsu [149], Bard [21]. In our opinion one of the most comprehensive classification of methods for BLP is presented in Colson et al. [49]. According to the authors, the majority of methods proposed for BLP are based on the following techniques:

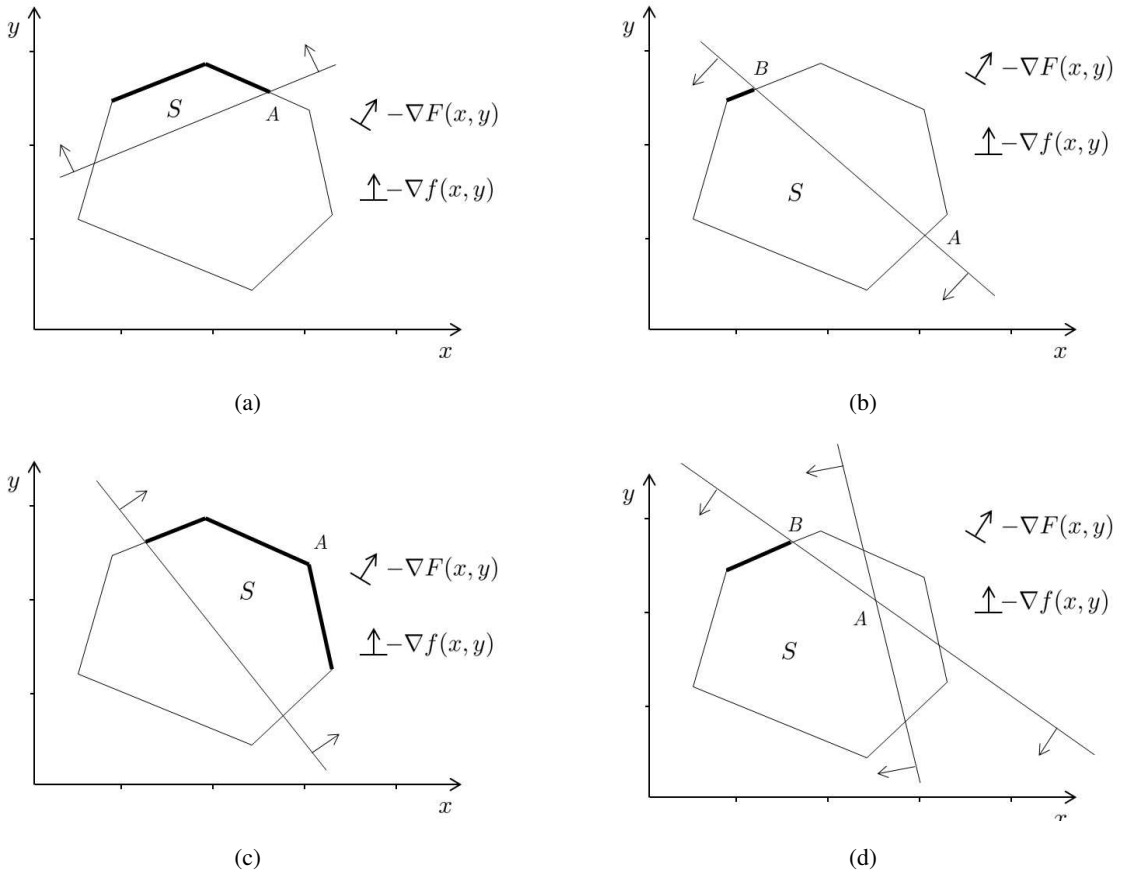


Figure 2.6. Single level relaxation with upper level constraints

- extreme points search
- reformulation
- complementary pivoting
- penalty function
- metaheuristics.

The methods based on extreme points search follows the same rationale of the classical simplex method exploiting the weak convex property of the inducible region. The basic framework of the simplex algorithm is properly modified to cope with the rationality requirement of a bilevel–feasible solution. One of the first method is the K^{th}-best proposed by Bialas and Karwan [34], other similar methods were proposed by Candler and Townsley [43] and Papavassilopoulos [121].

The reformulation methods is one of the most used due to its easy implementation. The idea is to replace the inner problem with its Karush–Kuhn–Tucker (KKT) conditions and then solve a single level problem. This approach, that was first introduced by Fortuny-Amat and McCarl [62] will be widely used in the rest of this dissertation: we refer to such reformulation as the single level reformulation of a BLP. Note that this approach is valid under the following assumption: a) the follower problem has objective function $f(x, y)$ and constraints $g(x, y)$ which are continuously differentiable and convex functions, b) set $\Omega_x(y)$ is regular $\forall x$ such that $(x, y) \in S$, c) the follower's global optimal solution is taken into account (Dempe and Dutta [55]). If the inner problem is a continuous linear problem and the optimistic approach is implemented, all the requirements are satisfied and the follower problem can be replaced by its KKT optimality conditions. The way in which the complementary slackness conditions are treated changes from one method to another, see for instance Bard and Falk [23] and Bard and Moore [24] or the branch and bound methods proposed by Hansen et al. [74] which is one of the best performing.

Complementary pivoting methods are closely related to the reformulation approaches, since the inner problem is replaced by its KKT conditions and the single level reformulation is treated as a complementarity problem. The method finds a solution which satisfies the complementary slackness conditions and gains a value of the upper level objective function less then or equal a parameter α; by iteratively reducing the value of α the algorithm finds a bilevel–feasible solution. The first method of this kind was proposed by Bialas and Karwan [34], but Ben-Ayed and Blair [27] showed that the algorithm may fail to converge to an optimal solution. A modified and convergent version of this algorithm was proposed by Jùdice and Faustino [79].

Penalty function methods tries to circumvent the bilevel structure of the problem solving a penalized single level version in which the duality gap of the inner problem is penalized in the leader's objective

function. Anandalingam and White [4] propose a penalty function method able to find a local optimal solution that was later improved to compute an optimal solution, see White and Anandalingam [152].

Finally, several metaheuristic approaches are proposed in the literature, for instance methods based on scalarization of leader's and follower's objective functions (Bard [18]), simulated annealing (Anandalingam et al. [3]), genetic algorithm (Mathieu et al. [110]) and tabu search (Gendreau et al. [68]).

2.4 Continuous and discrete BLPs

The general BLP framework presented so far features continuous upper level and lower level variables. For this class of bilevel problems there is a wide knowledge about geometrical properties and solution methods. We now address discrete bilevel linear problems that represents the main focus of the dissertation. A general classification of discrete bilevel problems is based on the nature of the leader's and follower's variables. Given variables $x \in X$ and $y \in Y$, we can define three different classes:

- Discrete–Continuous Bilevel Linear Problems (DCBLP), when $X = \mathbb{Z}^n$ and $Y = \mathbb{R}^m$
- Discrete Bilevel Linear Problems (DBLP), when $X = \mathbb{Z}^n$ and $Y = \mathbb{Z}^m$
- Continuous–Discrete Bilevel Linear Problems (CDBLP), when $X = \mathbb{R}^n$ and $Y = \mathbb{Z}^m$.

In the following chapters we study characteristics and solution methods for the first two classes. Indeed, DCBLPs are the most closely related to continuous BLPs and most of their polyhedral properties above described can be extended to the discrete–continuous case. In Chapter 3 we investigate two different solution approaches and in Chapter 5 we provide an application of a DCBLP model in the field of facility location. DBLPs represent pure discrete bilevel problems and their study is interesting in an applicative perspective as might allow to model a wide class of real integer problems joining their combinatorial nature and the bilevel structure. Despite the high potentiality and the great interest for DBLP, there is a lack of efficient methods, exact algorithms or fast heuristics, and this is certainly a promising direction of research. In Chapter 4 we investigate the geometrical properties of DBLPs both in a combinatorial and bilevel perspective, provide new theoretical results and solution algorithms and in Chapter 5 we describe an application for the Grid scheduling problem. Finally, CDBLPs are the less studied bilevel problems and are considered the hardest one. It is easy to understand that the complexity of bilevel problems strictly depends on the structure of the inner problem. Nevertheless, unlike DBLPs in which both the outer and the inner problems are integer, in CDBLPs the upper level variables are continuous and this hampers to use well known solution methods based on branch and bound and branch and cut. This class is out of the scope of this dissertation and will be object of future research activity.

Some preliminary theoretical results for discrete BLPs are the following.

Proposition 3. *(Vicente et al. [147]) Given a BLP and considering the three different classes of associated discrete problems:*

- *the inducible region of BLP contains the inducible region of DCBLP, i.e.* $IR_{DCLP} \subseteq IR_{BLP}$
- *the inducible region of CDBLP contains the inducible region of DBLP, i.e.* $IR_{DBLP} \subseteq IR_{CDBLP}$.

Proposition 4. *(Vicente et al. [147]) If* S *is bounded and non empty and there are no upper level constraints, BLP, DCBLP and DBLP admit an optimal solution if* $\{X \times Y\} \cap S \neq \emptyset$*, i.e. the inducible region is non empty. This is not true for CDBLP.*

From Proposition 3 it follows that all the polyhedral properties described for BLP, are also valid for DCBLP and that the optimal solution of BLP is a valid lower bound for the DCBLP. Otherwise, there is no any relationship between BLP and DBLP as the discrete nature of the inner problem deeply changes the reaction set and the inducible region of the problem. Proposition 4 states that under suitable conditions (S is bounded and $p = 0$), if the inducible region is non empty all problems but CDBLP has an optimal solution: this confirms that the latter is the hardest discrete bilevel problem from a computational standpoint. We show this phenomenon in the following example.

$$
\begin{aligned}
\min_{x,y}\ & x - 10y \\
\text{s.t.}\quad & x \in X \\
& y \in \operatorname*{argmin}_{y}\ y \\
& \quad \text{s.t.}\quad -x + 3y \leq 6 \\
& \qquad\quad x + y \leq 6 \\
& \qquad\quad x - 3y \leq 0 \\
& \qquad\quad y \geq 1 \\
& \qquad\quad x \geq 1 \\
& \qquad\quad y \in Y
\end{aligned}
$$

We can notice that, comparing Figures 2.7(a) and (b), the optimal solution of BLP is a lower bound of the optimal solution of DCBLP, while the optimal solution of DBLP is better than the previous two, despite the variables integrality. We describe more in detail these relationships in the following chapters. Finally, in Figure 2.7(d) we observe that, even if the inducible region is non empty, the CDBLP does not admit an optimal solution since point A is not rational.

2.5 Reformulation techniques

The relationship between continuous and discrete BLP arises more clearly when the integer variables are assumed to be binary. Starting from the results showed by Vicente et al. [147] and extended by

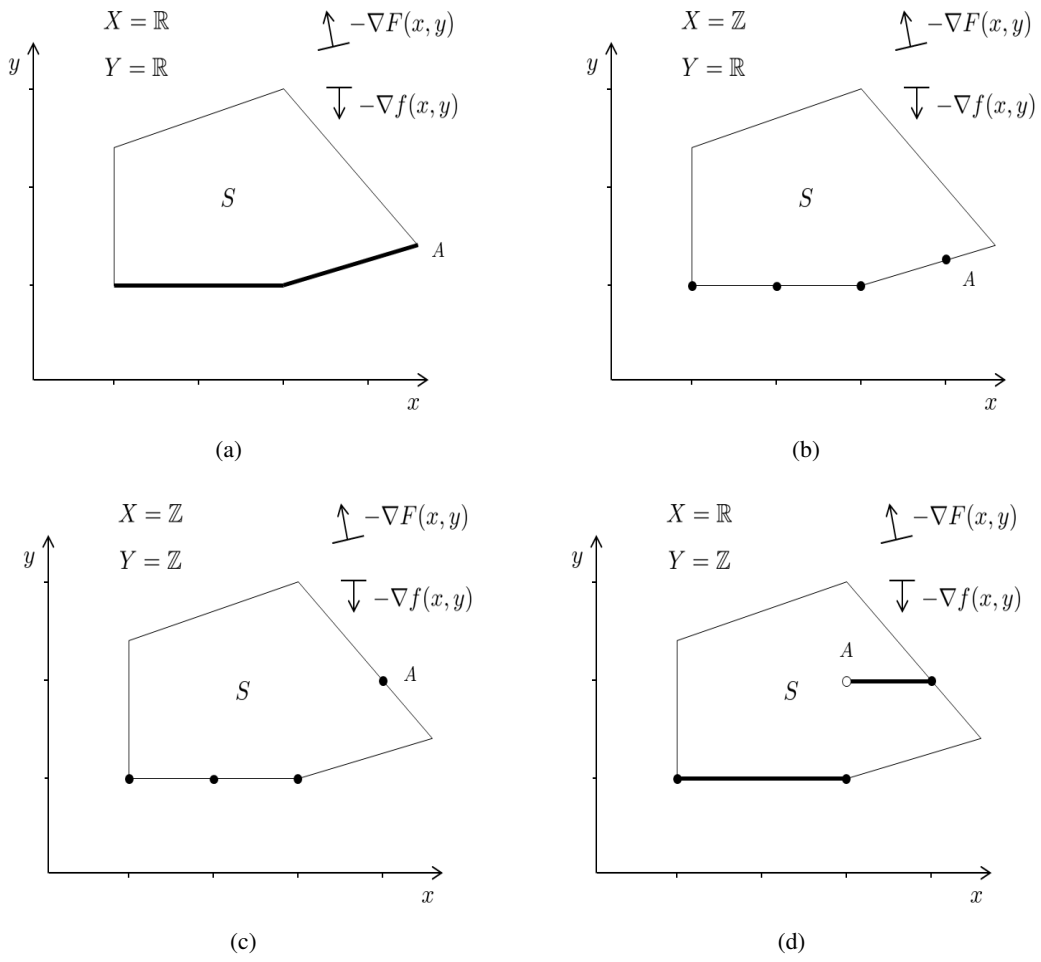


Figure 2.7. Continuous and Discrete Bilevel Linear Problems

Audet et al. [13], we describe some interesting reformulation techniques that prove the equivalence between 0-1 mixed-integer problems and BLPs.

Let us consider the following problems:

$$
\begin{aligned}
(DCBLP) \quad & \min_{x,y} c_1^T x + c_2^T y \\
& \text{s.t.} \quad Cx + Dy \leq e \\
& \qquad x \in \{0,1\} \\
& \qquad y \in \operatorname*{argmin}_{y} d^T y \\
& \qquad\quad \text{s.t.} \quad Ax + By \leq d \\
& \qquad\qquad y \geq 0
\end{aligned}
\qquad\qquad
\begin{aligned}
(DCBLP_z) \quad & \min_{x,y,z} c_1^T x + c_2^T y \\
& \text{s.t.} \quad Cx + Dy \leq e \\
& \qquad z = 0 \\
& \qquad 0 \leq x \leq 1 \\
& \qquad (z,y) \in \operatorname*{argmin}_{z,y} d^T y - \mathbb{1}^T z \\
& \qquad\quad \text{s.t.} \quad Ax + By \leq d \\
& \qquad\qquad z \leq x \\
& \qquad\qquad z \leq 1 - x \\
& \qquad\qquad y \geq 0
\end{aligned}
$$

Theorem 12. *(Audet et al. [13])* (x^*, y^*) *is an optimal solution of* $DCBLP$ *if and only if* (x^*, y^*, z^*) *is an optimal solution of* $DCBLP_z$ *with* $z^* = 0$.

In problem DCBLP$_z$ we introduce an auxiliary variable z that is defines as $z = \min\{x, 1 - x\}$. The upper level constraint $z = 0$ forces variable x to be binary. This suggests that an upper level binary variable x can be always relaxed by introducing an auxiliary follower variable z, a set of lower level constraints and a set of upper level equality constraints. The resulting formulation is still a bilevel problem with all continuous variables. As we previously noted, the presence of upper level constraints can induce infeasibility or discontinuity in the inducible region. For these reasons, especially when $p = 0$, may be convenient to use another reformulation approach. In Chapter 3 we describe a reformulation of a DCBLP with binary upper level constraints to a BLP without introducing additional upper level constraints, but modifying the leader's function through a suitable penalty function.

Let us address a bilevel problem with binary lower level variables.

$$
\begin{aligned}
(CDBLP) \quad & \min_{x,y} c_1^T x + c_2^T y \\
& \text{s.t.} \quad Cx + Dy \leq e \\
& \qquad x \geq 0 \\
& \qquad y \in \operatorname*{argmin}_{y} d^T y \\
& \qquad\quad \text{s.t.} \quad Ax + By \leq d \\
& \qquad\qquad y \in \{0,1\}
\end{aligned}
\qquad\qquad
\begin{aligned}
(CDBLP_z) \quad & \min_{x,y,z} c_1^T x + c_2^T y \\
& \text{s.t.} \quad Cx + Dy \leq e \\
& \qquad x \geq 0 \\
& \qquad y \in \operatorname*{argmin}_{y,z} d^T y \\
& \qquad\quad \text{s.t.} \quad Ax + By \leq d \\
& \qquad\qquad z = 0 \\
& \qquad\qquad 0 \leq y \leq 1 \\
& \qquad\qquad z \in \operatorname*{argmin}_{z} -\mathbb{1}^T z \\
& \qquad\qquad\quad \text{s.t.} \quad z \leq y \\
& \qquad\qquad\qquad z \leq 1 - y
\end{aligned}
$$

Theorem 13. *(Audet et al. [13])* (x^*, y^*) *is an optimal solution of* $CDBLP$ *if and only if* (x^*, y^*, z^*) *is an optimal solution of* $CDBLP_z$ *with* $z^* = 0$.

Theorem 13 states the equivalence between a continuous discrete bilevel problem and a three level problem in which the integrality requirements are relaxed. Clearly, this is only a theoretical result, but it shows the inherent link between binary linear problems and multilevel programming. We can now generalize the previous results.

Proposition 5. *Given an* $n-$*level problem with binary variables in the innermost level, it is possible to define an equivalent* $(n+1)-$*level problem with all continuous variables. If the binary variables do not belong to the innermost level, the equivalent continuous problem is still an* $n-$*level problem.*

This reformulation can be always applied to relax a binary variables, but the integrality relaxation requires the addition of an auxiliary lower level variable to the original formulation. Moreover, it is relevant to consider the level in which the binary variables are contained. If the bilevel problem has upper level binary variables, in the reformulated problem the auxiliary variables can be added to the existing lower level and it does not require a new level; if the binary variables are under control of the follower a new level has to be defined to contain the auxiliary variables. This explains why this reformulation approach is not used for binary DBLP and CDBLP: due to the integer nature of the lower level problem, the reformulation is a three level model for which efficient solution methods have not been developed yet.

Chapter 3

Discrete–Continuous Bilevel Linear Programming

Unlike continuous BLP, if at least one among the leader and follower variables are assumed to be integer, the resulting problem is much harder to tackle. Moore and Bard [114] are the first authors to point out the computational difficulties of integer bilevel linear problems and propose a branch and bound method for the more general mixed-integer case. In this chapter we address the first class of integer bilevel linear problems. We start from the discrete–continuous case in which the upper level variables are integer and the lower level variables are continuous, since this problem is the most closely related to continuous BLP. The main results we present in this chapter have been obtained following two different research direction: a) reformulation techniques, b) valid inequalities for DCBLP. In the following, these two area of research are treated separately to ease presentation.

3.1 Reformulation approaches for binary DCBLP

As we mentioned in the previous chapters, theoretical properties and possible reformulations are described in the literature in order to cope with the discrete nature of variables and exploit well known solution approaches for BLP.

In particular, in this section we focus on Discrete–Continuous Bilevel Linear Programs in which the upper level variables are discrete and the lower level variables are continuous. This problem is addressed, among others, by Wen and Yang [151], where a branch and bound approach and an heuristic are proposed, and by Wen and Huang [150], where the authors implement a tabu-search scheme to solve the problem.

One of the solution methods for this class of problems was proposed by Vicente et al. [147] and is based on the reformulation of the DCBLP as a continuous BLP. The authors generalize a result of Kalantary and Rosen [80] about 0-1 integer problems and extend the result to DCBLP. This

reformulation is based on the same rationale of the reformulation techniques presented in Chapter 2. Nevertheless, unlike the above mentioned, in this approach there are no additional upper level constraints, but the leader's objective function is modified by adding a concave penalty function weighted by a parameter μ.

The two problems have the same optimal solution for a sufficiently large value of μ. Vicente et al. in their paper proposed a lower bound for μ such that any value greater than this bound guarantees that the solution to the reformulated problem is integer. The effectiveness of the authors' proposal was not experimentally tested, and, therefore, the advantage of using a relatively small μ instead of a large one is not known. In the rest of this section this reformulation technique is investigated. We show the main criticality of this approach and then we try to improve the existing result by reducing the lower bound for parameter μ and proposing two new lower bounds. This represents a preliminary study to understand and assess whether the reduction of the penalty parameter may be considered a valuable line of research in order to develop a new efficient reformulation technique for DCBLP. Experiments on these new proposals are provided along with a comparison of the results obtained with the known bound.

3.1.1 Preliminary results

We address the following generic formulation P of a binary DCBLP:

$$
\begin{aligned}
(P) \quad \min_{x,y} \; & F(x,y) = -c_1^T x - c_2^T y \\
\text{s.t.} \quad & Cx + Dy \leq e \\
& x \in \{0,1\} \\
& y \in \operatorname*{argmin}_{y} f(y) = d^T y \\
& \qquad \text{s.t.} \quad Ax + By \leq b \\
& \qquad\qquad y \geq 0
\end{aligned}
$$

We can assume, without loss of generality, that $c_1 \geq 0$ and $c_2 \geq 0$. Now consider the continuous relaxation P_0 of problem P

$$
\begin{aligned}
(P_0) \quad \min_{x,y} \; & F(x,y) = -c_1^T x - c_2^T y \\
\text{s.t.} \quad & Cx + Dy \leq e \\
& 0 \leq x \leq 1 \\
& y \in \operatorname*{argmin}_{y} f(y) = d^T y \\
& \qquad \text{s.t.} \quad Ax + By \leq b \\
& \qquad\qquad y \geq 0
\end{aligned}
$$

in which the integrality constraints on variables x were relaxed. Let (x_0, y_0) be an optimal solution of the relaxed problem P_0. The inducible region IR_0 of P_0 is

$$IR_0 = \{(x,y) \mid 0 \leq x \leq 1, Cx + Dy \leq e, y \in \underset{y}{\text{argmin}}\{d^T y \text{ s.t. } By \leq b - Ax, y \geq 0\}\}.$$

It is known that for a sufficiently large value of a positive parameter μ problem P has the same optimal solution of the following parameterized problem:

$$\begin{aligned}
(P_\mu) \quad & \min_{x,y} F(x,y) = -c_1^T x - c_2^T y + \mu\, g(x) \\
& \text{s.t.} \quad Cx + Dy \leq e \\
& \qquad 0 \leq x \leq 1 \\
& \qquad y \in \underset{y}{\text{argmin}}\ f(y) = d^T y \\
& \qquad\qquad \text{s.t.} \quad Ax + By \leq b \\
& \qquad\qquad\qquad y \geq 0
\end{aligned}$$

where $g(x)$ is a continuous and concave function defined as follows:

$$\begin{aligned}
& g(x): \quad \mathbb{R}^n \to \mathbb{R} \\
& g(x): \quad \begin{cases} > 0 & \text{if} \quad 0 < x < 1 \\ = 0 & \text{if} \quad x \in \{0,1\} \end{cases}
\end{aligned}$$

Note that the set of feasible solutions of problem P_μ is the same of problem P_0, that is IR_0, because the two problems have different objective functions but are defined on the same set of constraints.

There are different functions $g(x)$ which satisfy the above requirements, but the one proposed in Vicente et al. is

$$g(x) = \sum_{i=1}^{n} \min[x_i\,;\,(1 - x_i)]$$

which is a piecewise linear function for $0 \leq x \leq 1$. The advantage of such a function is that, exploiting its piecewise linearity, P_μ can be formulated as a linear bilevel problem as explained in Chapter 2. Let us introduce a vector z of auxiliary variables z_i with

$$z_i = \min[x_i\,;\,(1 - x_i)] \qquad\qquad \forall i = 1...n$$

We can reformulate problem P_μ as follows:

$$
\begin{aligned}
(P_\mu) \quad \min_{x,y} \; & F(x,y) = -c_1^T x - c_2^T y + \mu \mathbb{1}^T z \\
\text{s.t.} \quad & Cx + Dy \leq e \\
& 0 \leq x \leq 1 \\
& y \in \operatorname*{argmin}_y f(y) = d^T y - \mathbb{1}^T z \\
& \qquad \text{s.t.} \quad Ax + By \leq b \\
& \qquad\qquad z \leq x \\
& \qquad\qquad z \leq 1 - x \\
& \qquad\qquad y \geq 0
\end{aligned}
$$

and the problem obtained is a continuous BLP. This reformulation shows that it is possible to relax the integrality requirements of the DCBLP modifying the problem but preserving the linearity of the objective function.

Vicente et al. proved that there exists a finite value μ_0 which is a lower bound for parameter μ. It means that for every $\mu > \mu_0$, an optimal solution to problem P_μ is also an optimal solution to problem P. Now we show how this lower bound μ_0 can be computed. Let us define the set of solutions of problem P_0 in which the x variables are not binary, that is

$$\overline{IR} = IR_0 \setminus \{x_i = 0 \vee x_i = 1, i = 1...n\}.$$

Let $\bar{x}$ be an optimal solution to the problem

$$
\begin{aligned}
\min_x \; & g(x) \\
\text{s.t.} \quad & x \in \overline{IR}
\end{aligned}
$$

Note that solving this problem on the feasible set IR_0 is trivial because the optimal solution would have $\bar{x}_i = 0$ or $\bar{x}_i = 1 \; \forall i = 1...n$, hence $g(\bar{x}) = 0$. Therefore, as $\bar{x} \in \overline{IR}$, $g(\bar{x}) > 0$ by definition of function $g(x)$. The lower bound μ_0 is

$$\mu_0 = \frac{c_1^T x_0 + c_2^T y_0}{g(\bar{x})}$$

which is a positive constant. From the latter formula, it is clear that the parameter is defined only if the denominator is not zero, that is the reason why the function $g(x)$ is minimized over $\overline{IR}$.

In the following section we show that μ_0 is a valid lower bound of μ providing an alternative proof than the one proposed in Vicente et al. The computation of μ_0 requires the solution of two problems, the bilevel linear problem P_0 and the minimization of the concave function $g(x)$.

3.1.2 Lower bound improvements

In order to present our first lower bound, we need to introduce another problem strictly related to the continuous relaxation P_0. Let problem P_0^{max} be defined as follows

$$
\begin{aligned}
(P_0^{max}) \quad & \max_{x,y} F(x,y) = -c_1^T x - c_2^T y \\
& \text{s.t.} \quad Cx + Dy \leq e \\
& \qquad 0 \leq x \leq 1 \\
& \qquad y \in \operatorname*{argmin}_y f(y) = d^T y \\
& \qquad\qquad \text{s.t.} \quad Ax + By \leq b \\
& \qquad\qquad\qquad y \geq 0
\end{aligned}
$$

which differs from problem P_0 because is a maximization problem defined on the same feasible set IR_0. Let (x_0^{max}, y_0^{max}) be an optimal solution of problem P_0^{max}. It is possible to define a better lower bound for parameter μ as follows

$$\mu' = \frac{c_1^T x_0 + c_2^T y_0 - c_1^T x_0^{max} - c_2^T y_0^{max}}{g(\bar{x})}$$

Proposition 6. *Parameter $\mu' \leq \mu_0$.*

Proof. P_0 and P_0^{max} are defined on the same feasible set IR_0 and a generic solution $(x, y) \in IR_0$ is such that $(x, y) \geq 0$. Since $c_1 \geq 0$ and $c_2 \geq 0$, then

$$-c_1^T x - c_2^T y \leq -c_1^T x_0^{max} - c_2^T y_0^{max} \leq 0 \qquad \forall (x,y) \in IR_0$$

Hence, we have

$$\mu' - \mu_0 = \frac{-c_1^T x_0^{max} - c_2^T y_0^{max}}{g(\bar{x})} \leq 0$$

Note that if $0 \in IR_0$, then $(x_0^{max}, y_0^{max}) = 0$ and $\mu' = \mu_0$. In this case the lower bound we define is not strictly less than the one proposed in by Vicente et al. □

We now provide a constructive proof of the validity of parameter μ'.

Theorem 14. *For any positive parameter $\mu > \mu'$ BLP problem P_μ and DCBLP problem P have the same optimal solution.*

Proof. Let $(\tilde{x}, \tilde{y})$ be an optimal solution of problem P_μ. If $(\tilde{x}, \tilde{y}) \in IR$, then $g(\tilde{x}) = 0$, $(\tilde{x}, \tilde{y})$ is an optimal solution of problem P and the theorem is trivially demonstrated.

Now assume that $(\tilde{x}, \tilde{y}) \notin IR$, i.e. $\tilde{x} \notin \{0,1\}^n$. Let (x, y) be a generic solution of P with variables x having binary components.

$$F(x,y) - F(\tilde{x}, \tilde{y}) = -c_1^T x - c_2^T y + c_1^T \tilde{x} + c_2^T \tilde{y} - \mu\, g(\tilde{x})$$

Since $\mu > \frac{c_1^T x_0 + c_2^T y_0 - c_1^T x_0^{max} - c_2^T y_0^{max}}{g(\bar{x})}$, we have

$$\begin{aligned} F(x,y) - F(\tilde{x},\tilde{y}) &= -c_1^T x - c_2^T y + c_1^T \tilde{x} + c_2^T \tilde{y} - \mu\, g(\tilde{x}) \\ &\leq -c_1^T x - c_2^T y + c_1^T \tilde{x} + c_2^T \tilde{y} - \mu\, g(\bar{x}) \\ &< c_1^T(\tilde{x} - x_0) + c_2^T(\tilde{y} - y_0) + c_1^T(x_0^{max} - x) + c_2^T(y_0^{max} - y) \end{aligned}$$

Recall that P_μ and P_0 are defined on the same feasible set IR_0 and that the feasible set of P is such that $IR \subseteq IR_0$, thus the following relationships hold

$$\begin{aligned} -c_1^T x_0 - c_2^T y_0 &\leq -c_1^T \tilde{x} - c_2^T \tilde{y} && \text{as } (\tilde{x},\tilde{y}) \text{ is a feasible solution of } P_0 \\ -c_1^T x - c_2^T y &\leq -c_1^T x_0^{max} - c_2^T y_0^{max} && \text{as } (x,y) \text{ is a feasible solution of } P_0^{max} \end{aligned}$$

It follows that

$$F(x,y) - F(\tilde{x},\tilde{y}) < c_1^T(\tilde{x} - x_0) + c_2^T(\tilde{y} - y_0) + c_1^T(x_0^{max} - x) + c_2^T(y_0^{max} - y) \leq 0$$

hence

$$F(x,y) - F(\tilde{x},\tilde{y}) < 0$$

which contradicts the assumption that $(\tilde{x},\tilde{y})$ is an optimal solution of P_μ. Thus, $(\tilde{x},\tilde{y}) \in IR$, i.e. $\tilde{x} \in \{0,1\}^n$, and we have

$$F(\tilde{x},\tilde{y}) \leq F(x,y) \qquad \forall (x,y) \in IR$$

which concludes the proof. □

In the proof of Theorem 14 we compute a sufficiently large value for the penalty parameter to assure that the best non integer solution $(\tilde{x},\tilde{y})$ of P_μ is such that $F(x,y) < F(\tilde{x},\tilde{y})$ for any feasible integer solution (x,y) of P. The same result may be obtained using a tighter bound.

In fact, it is enough to guarantee that $(\tilde{x},\tilde{y})$ gains a worse objective value than any *optimal* solution of P. This allows us to define a new valid lower bound for μ,

$$\mu'' = \frac{c_1^T x_0 + c_2^T y_0 - c_1^T x_P - c_2^T y_P}{g(\bar{x})}$$

where (x_P, y_P) is any feasible solution of P.

Proposition 7. *Parameter* $\mu'' \leq \mu' \leq \mu_0$.

Proof. The second inequality was already shown, while for the first one note that $IR \subseteq IR_0$, hence must hold

$$-c_1^T x_P - c_2^T y_P \leq -c_1^T x_0^{max} - c_2^T y_0^{max}$$

Thus

$$\mu'' - \mu' = \frac{-c_1^T x_P - c_2^T y_P + c_1^T x_0^{max} + c_2^T y_0^{max}}{g(\bar{x})} \leq 0$$

which completes the proof. □

Note that the difference among μ'' and the other two parameters depends on the quality of (x_P, y_P): if $F(x_P, y_P)$ is far from the optimum, μ'' is almost the same of μ' and μ_0. On the contrary, if $F(x_P, y_P)$ is close to the optimum, μ'' may be much smaller. Furthermore, for the computation of μ'' it is not necessary to solve to optimality problem P, but only a feasible solution is required. Hence, the computation of μ'' is not more time consuming than the computation of μ_0 and only depends on the algorithm used to compute a feasible solution of P.

Now we can formulate a modified version of Theorem 14.

Theorem 15. *For any positive parameter $\mu > \mu''$, the BLP problem P_μ and the DCBLP problem P have the same optimal solution.*

Proof. Let $(\tilde{x}, \tilde{y})$ be an optimal solution of problem P_μ and let (x^*, y^*) be an optimal solution of P. Following the same arguments of Theorem 14, if $(\tilde{x}, \tilde{y}) \in IR$, the theorem is trivially proved. Otherwise, if $(\tilde{x}, \tilde{y}) \notin IR$, we have

$$F(x^*, y^*) - F(\tilde{x}, \tilde{y}) = -c_1^T x^* - c_2^T y^* + c_1^T \tilde{x} + c_2^T \tilde{y} - \mu\, g(\tilde{x})$$

and by the definition of μ''

$$F(x^*, y^*) - F(\tilde{x}, \tilde{y}) < c_1^T(\tilde{x} - x_0) + c_2^T(\tilde{y} - y_0) + c_1^T(x_P - x^*) + c_2^T(y_P - y^*)$$

But now we have

$$-c_1^T x^* - c_2^T y^* \leq -c_1^T x_P - c_2^T y_P$$

It follows that

$$F(x^*, y^*) - F(\tilde{x}, \tilde{y}) < 0$$

i.e. there exists a feasible solution (x^*, y^*) for P_μ that is better than the optimal solution $(\tilde{x}, \tilde{y})$, which is a contradiction. Therefore, $(\tilde{x}, \tilde{y}) \in IR$, and we have

$$F(\tilde{x}, \tilde{y}) \leq F(x^*, y^*)$$

with (x^*, y^*) optimal solution of P, hence the proof is complete. □

This result implies that it is possible to reformulate the problem using tighter lower bounds for the penalty parameter, in order to reduce approximation errors and undesirable behaviors that may occur when commercial solvers are used to compute an optimal solution of the original binary DCBLP.

3.1.3 Experimental analysis

Implementation details

In this section we analyze the behavior of the lower bounds for μ proposed and compare them to the lower bound proposed by Vicente et al. We also experimented with a very large value for μ in order to underline the effectiveness of this penalty parameter in terms of CPU time reduction. Our main goal is to realize a preliminary study on this reformulation approach and to understand whether the reduction of the penalty parameter is a promising direction of research. Our intention is to assess the computational burden required to solve the bilevel problem according to the reduction of the penalty parameter μ: if the new lower bound provides a remarkable time reduction, it means that the computation of tighter lower bounds for μ may represent a possible area of future research in order to solve bigger size problems.

To this end we considered 17 different classes of problems, distinguished by the number of upper and lower level variables. For every class of problems we randomly generated 5 instances, yielding an overall number of 85 test problems. The values chosen for the leader and follower variables were $(n, m) \in \{5, 10, 15, 20\}$, and, among all the possible combinations of n and m, we solved a subset of the most significant ones. We set a fixed number $q = 2/5 \cdot (n + m)$ of lower level constraints and two possible values for the upper level constraints, i.e., $p = 0$ and $p = 1/5 \cdot (n + m)$. This choice was made in order to analyze the effect of the upper level constraints on the computational burden.

Following the experimental approach used in Bard [21], Wen and Huang [150] and Wen and Yang [151], the coefficients of matrices A_1, A_2, B_1, B_2 were randomly chosen in the range [-25,25], the coefficients of vectors b_1 and b_2 were randomly chosen between 0.4 and 0.8 times the sum of the row coefficients of the corresponding matrices, the (negative) coefficients of vectors c_1 and c_2 were randomly chosen in the range [-10,-1], and the coefficients of vector d_2 were randomly chosen between -5 and 5.

Table 3.1 shows the 17 classes of problems. All the problems were implemented in the AMPL language and optimally solved by means of the CPLEX 12.3 solver on a PC with a 2Ghz Pentium Core 2 Duo processor and 1GB of RAM.

For every class, we solved problems P_μ setting parameter μ to different values. The first comparison was made between the parameter μ_0 proposed by Vicente et al. and a very large value $\mu = 100\,000$. A further comparison was made among the previous parameters and the parameters we propose, μ' and μ''.

To compute all the lower bounds for μ, the denominator $g(\bar{x})$ was calculated solving the following mathematical program

class	(n, m)	(p, q)
1	(5,5)	(0,4)
2	(5,5)	(2,4)
3	(5,10)	(0,6)
4	(5,10)	(3,6)
5	(10,5)	(0,6)
6	(10,5)	(3,6)
7	(10,10)	(0,8)
8	(10,10)	(4,8)
9	(10,15)	(5,10)
10	(15,5)	(0,8)
11	(15,5)	(4,4)
12	(15,10)	(0,10)
13	(15,10)	(5,10)
14	(20,10)	(0,12)
15	(20,10)	(6,12)
16	(20,15)	(0,14)
17	(20,15)	(7,14)

Table 3.1. Classes of problems

$$
\begin{aligned}
&\min_{x,y,z} \mathbb{1}^T z \\
&\text{s.t.} \quad Cx + Dy \leq e \\
&\qquad 0 \leq x \leq 1 \\
&\qquad (y, z) \in \operatorname*{argmin}_{y,z} d^T y - \mathbb{1}^T z \\
&\qquad\qquad \text{s.t.} \quad Ax + By \leq b \\
&\qquad\qquad\qquad z \leq x \\
&\qquad\qquad\qquad z \leq 1 - x \\
&\qquad\qquad\qquad y \geq 0 \\
&\qquad\qquad\qquad x \geq \varepsilon - M\delta^1 && (1) \\
&\qquad\qquad\qquad x \leq M(1 - \delta^1) && (2) \\
&\qquad\qquad\qquad (1 - x) \geq \varepsilon - M\delta^2 && (3) \\
&\qquad\qquad\qquad (1 - x) \leq M(1 - \delta^2) && (4) \\
&\qquad\qquad\qquad \delta \leq \delta^1 + \delta^2 && (5) \\
&\qquad\qquad\qquad \delta \geq \frac{\delta^1 + \delta^2}{2} && (6) \\
&\qquad\qquad\qquad \Delta \geq \sum_{i=1}^{n} \delta_i - (n - 1) && (7) \\
&\qquad\qquad\qquad \Delta \leq \frac{\sum_{i=1}^{n} \delta_i}{n} && (8) \\
&\qquad\qquad\qquad \Delta \leq 0 && (9)
\end{aligned}
$$

Note that, unless the original problem P has not any integer solution, it is necessary to introduce suitable constraints to avoid that the optimal solution $\bar{x}$ has binary components. Constraints (1) and (2) model the following logical relation:

$$x \leq 0 \Leftrightarrow \delta^1 = 1$$

where M and ε are a sufficiently big and small constant respectively. Similarly, constraints (3) and (4) represent the following relation:

$$x \geq 1 \Leftrightarrow \delta^2 = 1$$

Variable δ is defined as $\delta = \delta^1 \vee \delta^2$, thus it is equal to 0 if and only if both δ^1 and δ^2 are 0, i.e. x is fractional. Variable Δ is defined as $\Delta = \wedge_{i=1}^{n} \delta_i$, thus it is equal to 0 if at least one δ_i is 0. Constraints (5)-(8) express the previous relations by means of well known linear constraints, and constraint (9) makes a non fractional solution infeasible.

The latter is a very time consuming problem to solve. The computational complexity does not depend on the penalty parameter we compute and for this reason it represents the most important deficiency for this reformulation pattern so far. It is clear that solving this problem efficiently is an obliged direction for future research in order to make this approach applicable to real life instances. As far as the computational analysis is concerned we set a time limit of 5000 seconds to compute $\bar{x}$. The time limit was reached only for 13 instances out of the 95 (13.7%). Note that this computational time is the same regardless of the penalty parameter used and it is often much more than the time required to solve problem P_μ. For this reason, in order to ease the comparison among the different parameters, the time required to compute $\bar{x}$ is not comprised in the CPU times recorded in the following. The same time limit of 5000 seconds was set for the solution of P_μ.

Problems P_0 and P_μ were solved replacing the follower problem by its KKT conditions according to the well known method of Fortuny-Amat and McCarl [62]. To compute μ'', a feasible solution was obtained using the following heuristic procedure. We first solved the single level version of P removing the lower level objective function (single level relaxation). Next, according to the solution found, we fixed the values of the upper level variables and found a bilevel–feasible solution. If the single level relaxation was unbounded or its solution did not satisfy the upper level constraints, we randomly generated a vector of upper level variables and then a bilevel–feasible solution was calculated.

Finally we note that only instances for which problem P_0 did not have any integer optimal solution have been considered. Such an instance may be trivially computed solving problem P_μ with $\mu = 0$, and, therefore, it is meaningless comparing computational results obtained for different values of μ.

class	$\mu = 100\,000$	μ_0	
	CPU	*CPU*	ΔT
1	0.09	0.08	-0.02
2	0.09	0.07	-0.02
3	0.28	0.18	-0.10
4	0.15	0.15	0.00
5	0.52	0.55	0.03
6	0.43	0.22	-0.21
7	11.23	4.86	-6.37
8	2.76	3.10	0.34
9	18.38	12.63	-5.75
10	62.42	53.92	-8.50
11	2.75	2.54	-0.21
12	7.95	7.33	-0.62
13	89.30	72.04	-17.26
14	1455.86	1236.61	-219.25
15	1656.64	1637.05	-19.59
16	3152.16	3058.62	-93.54
17	2346.58	1908.61	-437.98

Table 3.2. Computational comparison between a very large penalty parameter $\mu = 100\,000$ and μ_0

Computational comparison

We compared the CPU time obtained with μ_0 and $\mu = 100\,000$ to quantify the connection between the value of the penalty parameter μ and the required computational effort. This very large value of μ was chosen to guarantee the integrality of the solution computed by solving problem P_μ. The average CPU time for every class of instances are shown in Table 3.2.

We note that for the largest instances, from class 13 to 17, the computational burden required with parameter μ_0 is significantly less. In general, we can observe a clear and constant result because μ_0 outperforms a large penalty parameter $\mu = 100\,000$ with exception for some classes of small size problems in which the two CPU times consumed are almost the same (see, e.g., classes 4, 5 or 8).

If we make the same comparison for μ' and μ'' we obtain the results summarized in Table 3.3.

Now it is possible to observe that the major differences in computational time occur when the problem size grows. The general performance obtained with parameters μ' and μ'' are quite similar to the one obtained with μ_0: the CPU time consumed, compared to the case with $\mu = 100\,000$, is quite similar for small and medium size problems and significantly less for large size problems. We note the largest improvement with μ'' for class 17. The average values of the above mentioned comparison are showed in Table 3.4.

class	$\mu = 100\,000$	μ'		μ''	
	CPU	*CPU*	ΔT	*CPU*	ΔT
1	0.09	0.07	-0.02	0.07	-0.03
2	0.09	0.06	-0.03	0.06	-0.02
3	0.28	0.16	-0.11	0.16	-0.12
4	0.15	0.14	-0.01	0.13	-0.02
5	0.52	0.56	0.04	0.52	0.01
6	0.43	0.23	-0.20	0.23	-0.20
7	11.23	4.86	-6.37	4.90	-6.33
8	2.76	3.59	0.83	2.78	0.02
9	18.38	12.44	-5,94	10.48	-7.90
10	62.42	59.19	-3.23	48.32	-14.10
11	2.75	3.15	0.40	3.03	0.28
12	7.95	5.47	-2.48	4.45	-3.50
13	89.30	71.83	-17.48	49.75	-39.56
14	1455.86	1226.37	-229.49	1216.30	-239.56
15	1656.64	1641.45	-15.19	1189.00	-467.64
16	3152.16	2628.25	-523.91	3046.11	-106.05
17	2346.58	1803.55	-543.04	969.37	-1377.21

Table 3.3. Computational comparison among a very large penalty parameter $\mu = 100\,000$, μ' and μ''

$\mu = 100\,000$	μ_0		μ'		μ''	
CPU	*CPU*	ΔT	*CPU*	ΔT	*CPU*	ΔT
518.09	470.50	-47.59	438.90	-79.19	385.04	-133.05

Table 3.4. Comparison of average values

In order to make a more complete comparison among the parameter proposed by Vicente et al. and the proposed ones, in the following, we report a series of detailed data computed for all the instances solved.

The main features we recorded for every run are:

- Δ_μ, the percentage difference among parameters μ_0 and μ' or μ''
- *CPU*, the CPU time (in seconds) spent to solve the problem
- ΔT, the difference between the CPU time spent to solve P_{μ_0} and $P_{\mu'}$ or $P_{\mu''}$
- n^o *better*, the number of times the time spent to solve $P_{\mu'}$ or $P_{\mu''}$was less than or equal to the time spent to solve P_{μ_0}
- *iter*, the number of MIP simplex iterations to solve the problem.

In Table 3.5 we report the computational results.

First of all we note that the numerical difference between μ_0 and μ' ranges from -5.40% to -41.84% and it is equal to -20.39% on average and the difference between μ_0 and μ'' ranges from -13.70% to -86.84%, with an average value of -66.38%. Recall that the value of μ'' strongly depends on the heuristic algorithm used to determine a good feasible solution. Table 3.5 shows that the reduction of parameter μ produces a time reduction for the solution of the problems.

If we compare parameters μ_0 and μ'' we can observe a clear and constant behavior as μ'' always allows to obtain a lower CPU time or a quite similar one. The overall comparison shows that μ'' ensures the best computational performance in 13 out of the 17 classes. The average values for *CPU*, n^o *better* and *simplex* are reported in Table 3.6.

In Table 3.6 we can see that the computational time obtained by using parameter μ' is not worse than the one obtained with μ_0 in 65% of the instances and this percentage grows to 80% using μ''. The parameters we propose ensure an average CPU time reduction of 6.72% and 18.16 % despite a slight increase in simplex iterations that occurs for μ'. Parameter μ'' is clearly the best performing one. All these results confirm and extend what was already pointed out comparing the parameters to $\mu = 100\,000$.

3.1.4 Conclusion

We proposed two improved lower bounds for parameter μ used to reformulate a binary DCBLP as a BLP. This approach was described in the literature, but only from a theoretical point of view. Our main purpose was to investigate the main criticalities of this reformulation scheme and to provide a preliminary insight on the effectiveness of the reduction of such penalty parameters. In fact, even if the method has been theoretically described in the literature, its practical implementation is omitted and

class	μ_0		μ'					μ''				
	CPU	*iter*	Δ_μ	*CPU*	ΔT	n^o *better*	*iter*	Δ_μ	*CPU*	ΔT	n^o *better*	*iter*
1	0,08	768	-19,41 %	0,07	-0,01	100 %	769	-63,54 %	0,07	-0,01	100 %	748
2	0,07	421	-40,89 %	0,06	-0,01	100 %	441	-62,72 %	0,06	0,00	80 %	507
3	0,18	3168	-32,80 %	0,16	-0,02	80 %	3056	-67,71 %	0,16	-0,02	100 %	2866
4	0,15	2560	-41,84 %	0,14	-0,01	80 %	2460	-63,55 %	0,13	-0,02	100 %	2268
5	0.55	10 924	-18.88 %	0.56	0.01	60 %	10 966	-69.01 %	0.52	-0.03	80 %	10 685
6	0.22	3647	-13.92 %	0.23	0.01	60 %	3743	-64.85 %	0.23	0.01	60 %	3759
7	4.86	106 849	-14.11 %	4.86	0.00	80 %	109 470	-64.95 %	4.90	0.04	40 %	118 832
8	3.10	67 413	-33.09 %	3.59	0.49	20 %	78 557	-72.14 %	2.78	-0.32	80 %	60 291
9	12.63	335 323	-24.90 %	12.44	-0.19	40 %	329 310	-86.84 %	10.48	-2.16	80 %	254 638
10	53.92	1 126 371	-8.77 %	59.19	5.27	20 %	1 410 454	-65.24 %	48.32	-5.60	60 %	1 093 042
11	2.54	54 134	-14.42 %	3.15	0.61	80 %	65 435	-76.93 %	3.03	0.49	60 %	64 119
12	7.33	159 883	-19.52 %	5.47	-1.85	80 %	114 971	-69.68 %	4.45	-2.87	60 %	87 737
13	72.04	2 042 327	-16.91 %	71.83	-0.21	60 %	1 877 718	-67.92 %	49.75	-22.29	60 %	1 154 466
14	1 236,61	32 050 869	-7.20 %	1226.37	-10.24	60 %	32 005 678	-65.77 %	1216.30	-20.31	100 %	32 691 768
15	1637.05	38 685 697	-13.82 %	1641.45	4.40	60 %	36 249 418	-74.52 %	1189.00	-448.05	100 %	25 980 097
16	3058.62	84 659 188	-5.40 %	2628.25	-430.37	100 %	77 524 263	-13.70 %	3046.11	-12.52	100 %	79 143 806
17	1908.61	24 714 769	-20.80 %	1803.55	-105.06	25 %	38 426 480	-79.35 %	969.37	-939.23	100 %	20 950 294

Table 3.5. Computational results of all the classes with μ_0, μ' and μ''

μ_0		μ'			μ''		
CPU	*iter*	*CPU*	n^o *better*	*iter*	*CPU*	n^o *better*	*iter*
470.50	10 824 960	438.90	65 %	11 071 364	385.40	80 %	9 507 054
vs μ_0		-6.72%	-	2.28 %	-18.16 %	-	-12.17%

Table 3.6. Average results for all the problems solved

the potential advantage of using a smaller penalty parameter is not investigated. Our first contribution was to provide a preliminary experimental test on this reformulation approach comparing the existing penalty parameter to a sufficiently large one. The results clearly show that the computational burden is affected by the size of the parameter used. The second contribution was to propose a new theoretical result providing two reduced lower bounds for penalty parameter and comparing them to the existing one. Hence, the reformulation approach was investigated from both a theoretical and a computational perspective.

More in detail, computational results show that the bounds proposed are effective, with an average CPU time reduction of 6% and 18%. This result is also strengthened if we compare the CPU time gained by all the parameters and the one obtained with a very large $\mu = 100\,000$. In this case the average time reduction is -9% for μ_0, -15% for μ' and -25% for μ''. This implies that it may be possible to solve bigger instances of the problem and increase the applicability of integer bilevel programming to real problems.

Actually, the major deficiency of this reformulation approach, which still limits its applicability, is the computation of $g(\bar{x})$ which represents also the main drawback of our experimental analysis. This suggests that it is necessary to focus on $g(x)$ to compute $\bar{x}$ more easily. Future research work may be addressed following two different directions. On one side, it may be useful to provide other functions $g(x)$, always piecewise linear, whose computation is less hard, for example defining the same function on a larger number of subsets of the $[0, 1]$ interval. On the other side, further investigation will be focused on the definition of new penalty parameters in which the optimal solution $g(\bar{x})$ may be replaced by an upper bound of it in order to provide a tighter lower bound for μ with a smaller computational effort.

3.2 A New valid inequality for DBLP

3.2.1 Introduction

In this section we extend the study of DBLP assuming that the upper level variables are not binary, but integer in general. Starting from the computational difficulties previously mentioned, we propose a new valid inequality for DCBLPs. This valid inequality exploits the bilevel nature of the problem and

the geometry of the solutions space, trying to remove from the feasible region those solutions which are not bilevel–feasible. One idea stems from well known results for BLPs present in the literature.

This inequality can be used to reformulate the original problem in order to reduce the feasible region and speed up the resolution. The proposed valid inequality is added to the original problem. The optimal solution of the resulting DCBLP is computed by solving its single level reformulation.

We address the following generic formulation of a DCBLP:

$$\begin{aligned}
&\min_{x,y} F(x,y) = c_1^T x + c_2^T y \\
&\text{s.t.} \quad Cx + Dy \leq e \\
&\qquad x \in \mathbb{Z}_+ \\
&\qquad y \in \underset{y}{\text{argmin}}\ f(y) = d^T y \\
&\qquad\quad \text{s.t.} \quad Ax + By \leq b \\
&\qquad\qquad\quad y \geq 0
\end{aligned}$$

Recall the geometric properties 7 and 3 we presented in Chapter 2: a) the inducible region of a BLP can be written as a piecewise linear equality constraint and the latter is comprised of the supporting hyperplanes of S (Bard [20]), b) the inducible region of a DBLP is contained in the inducible region of the corresponding BLP obtained relaxing the integrality requirements on the leader's variables ([147]). These geometric properties are fundamental for the generation of the valid inequality described in the next section that is able to significantly reduce the size of feasible region S.

3.2.2 The continuous case: BLP

It is possible to obtain a relaxation of the inducible region of a DCBLP just considering the inducible region of a corresponding BLP in which the integrality requirements are dropped. For the sake of clearness, let us consider the following example. Let DCBLP be the original problem and BLP the associated continuous relaxation in which the integrality requirements are removed::

$$\begin{aligned}
&(DCBLP) \quad \min_{x,y} F(x,y) = x - 10y \\
&\qquad \text{s.t.} \quad 5x - 14y \leq -1 \\
&\qquad\quad x \in \mathbb{Z}_+ \\
&\qquad\quad y \in \underset{y}{\text{argmin}}\ f(y) = 3y \\
&\qquad\qquad \text{s.t.} \quad 3x + 2y \leq 20 \\
&\qquad\qquad\quad 5x - 2y \geq 2 \\
&\qquad\qquad\quad x - 3y \geq -9 \\
&\qquad\qquad\quad 2x - y \geq 16 \\
&\qquad\qquad\quad x + 8y \geq 11 \\
&\qquad\qquad\quad y \geq 0
\end{aligned}$$

$$\begin{aligned}
&(BLP) \quad \min_{x,y} F(x,y) = x - 10y \\
&\qquad \text{s.t.} \quad 5x - 14y \leq -1 \\
&\qquad\quad x \geq 0 \\
&\qquad\quad y \in \underset{y}{\text{argmin}}\ f(y) = 3y \\
&\qquad\qquad \text{s.t.} \quad 3x + 2y \leq 20 \\
&\qquad\qquad\quad 5x - 2y \geq 2 \\
&\qquad\qquad\quad x - 3y \geq -9 \\
&\qquad\qquad\quad 2x - y \geq 16 \\
&\qquad\qquad\quad x + 8y \geq 11 \\
&\qquad\qquad\quad y \geq 0
\end{aligned}$$

In Figure 3.1 the two inducible regions are shown. Let us denote with IR_{DCBLP} and IR_{BLP} the inducible region for problems $DCBLP$ and BLP respectively. IR_{DCBLP} is comprised of the black points, while the white points are rational solutions that violate the upper level constraints. IR_{BLP} is represented by the bold line and note that it is comprised of two supporting hyperplanes of S.

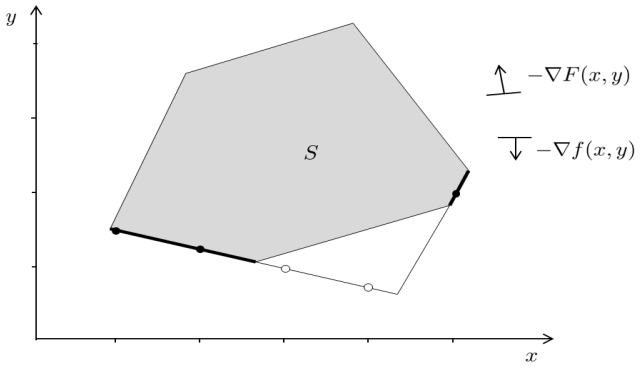


Figure 3.1. Comparison between two inducible regions

It holds that $IR_{DCBLP} \subseteq IR_{BLP}$. The optimal solution of DCBLP is $(x^*, y^*) = (5, 2)$ with $F(x^*, y^*) = -15$, while the optimal solution of BLP is $(x, y) \approx (5.1, 2.3)$ with $F(x, y) \approx -17.7 < F(x^*, y^*)$.

A BLP is more tractable than a DCBLP since the number of discrete variables is reduced and it is easy to define a halfspace containing all the bilevel–feasible solutions. The main objective is to cut off the part of the feasible region S comprised of solutions which are not bilevel–feasible and can be discarded without a bilevel–feasibility check. Let us define the following bilevel linear problem BLP_{min}^{max}

$$
\begin{aligned}
(BLP_{min}^{max}) \quad & \max_{x,y} f(y) = d^T y \\
& \text{s.t.} \quad Cx + Dy \leq e \\
& \qquad x \geq 0 \\
& \qquad y \in \operatorname*{argmin}_{y} f(y) = d^T y \\
& \qquad \text{s.t.} \quad Ax + By \leq b \\
& \qquad\qquad y \geq 0
\end{aligned}
$$

Let $(\hat{x}, \hat{y})$ be the optimal solution. Note that the leader's and the follower's objective functions are the same, but the first is maximized and the second is minimized. Hence, $(\hat{x}, \hat{y})$ represents the worst bilevel solution from the follower's perspective. The following proposition is verified.

Proposition 8. *Let $(\hat{x}, \hat{y})$ be the optimal solution of BLP_{min}^{max}. For a generic BLP the inequality $d^T y \leq d^T \hat{y}$ is a valid cut.*

Proof. If there exists a bilevel–feasible solution $(\bar{x}, \bar{y})$ such that $d^T\bar{y} > d^T\hat{y}$, it follows that there is a bilevel–feasible solution which is worse than $(\hat{x}, \hat{y})$ for the follower, which contradicts the formulation of problem BLP_{min}^{max}. It implies that $(\bar{x}, \bar{y}) \notin R_y(\bar{x})$, thus $(\bar{x}, \bar{y}) \notin IR$, hence the proof. □

Note that Proposition 8 holds regardless the presence of upper level constraints. The cut presented in Proposition 8 is only a theoretical result for BLPs. Even if it represents a valid inequality for a BLP, it requires the solution of BLP_{min}^{max} which is another BLP: it means that the computation of a valid cut for a problem is as difficult as the problem itself. Notwithstanding, Proposition 8 suggests an idea that can be somehow preserved in the discrete–continuous case. Since a DCBLP is harder than BLP_{min}^{max}, the computational burden required to solve BLP_{min}^{max} may be counterbalanced by a time reduction in solving DCBLP once the valid inequality is added.

3.2.3 The discrete–continuous case: DCBLP

It is easy to show the following result:

Proposition 9. *Let $(\hat{x}, \hat{y})$ be the optimal solution of BLP_{min}^{max}. For the DCBLP obtained by the BLP in which the leader's variables are required to be integer, the inequality $d^T y \leq d^T\hat{y}$ is a valid cut.*

Proof. Recall that $IR_{DCBLP} \subseteq IR_{BLP}$. This trivially implies that if an inequality is satisfied by all solutions in IR_{BLP}, the same holds for solutions in IR_{DCBLP}, hence the proof. □

The valid inequality described provides an upper bound on the value of the lower level objective function: all the solutions which yield a higher value, can not belong to the follower's reaction set nor to the inducible region and can be discarded. See Figure 3.2 for a graphic explanation of the valid inequality applied to the previous example.

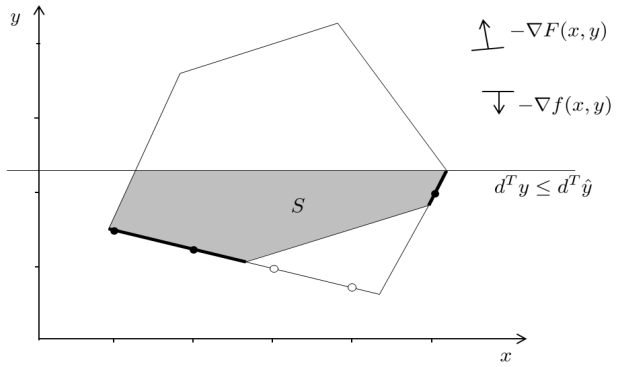


Figure 3.2. An application of the valid cut

It is important to note that the valid inequality can be used both like an upper and lower level constraint. Let us consider the two interpretations separately. If the valid inequality is treated like a lower level constraint, for each value of x the size of the reaction set $R_y(x)$ is reduced, but, on the other hand,

we increase the number of constraints of the follower problem hence its hardness. Let us consider the reformulation technique proposed by Fortuny-Amat and McCarl. Given a vector x of upper level variables the follower problem is replaced by its KKT conditions and the resulting set of constraints is the following:

$$\begin{aligned} Ax + By - b + t &= 0 & y &\leq M \cdot \beta \\ d + B^T\mu - s &= 0 & s &\leq M \cdot (1-\beta) \\ \mu &\leq M \cdot \alpha & \mu,\, t &\in \mathbb{R}^q_+,\ \alpha \in \mathbb{B}^q \\ t &\leq M \cdot (1-\alpha) & y,\, s &\in \mathbb{R}^m_+,\ \beta \in \mathbb{B}^m \end{aligned}$$

Binary variables α and β are necessary for linearizing the bilinear complementary slackness constraints by using a big-M approach. In this case, adding a constraint to the follower, we need also to increase the number of binary variables and duality variables. The resulting reformulation changes as follows:

$$\begin{aligned} Ax + By - b + t &= 0 & s &\leq M \cdot (1-\beta) \\ d^T y - d^T\hat{y} + v &= 0 & \varphi &\leq M \cdot \gamma \\ d + B^T\mu + d\varphi - s &= 0 & v &\leq M \cdot (1-\gamma) \\ \mu &\leq M \cdot \alpha & \mu,\, t &\in \mathbb{R}^q_+,\ \alpha \in \mathbb{B}^q \\ t &\leq M \cdot (1-\alpha) & y,\, s &\in \mathbb{R}^m_+,\ \beta \in \mathbb{B}^m \\ y &\leq M \cdot \beta & \varphi, v &\in \mathbb{R}_+,\ \gamma \in \mathbb{B} \end{aligned}$$

If the valid inequality is considered as an upper level constraint, the follower problem's size does not change, but we introduce a leader's constraint which intersects the reaction set in a small subset of points by construction. In other words, the resulting formulation is less hard to tackle, but the valid inequality may be less effective.

3.2.4 Computational results

In this section we assess the effect of the proposed valid inequality in terms of computational performance. Starting from an instance of a DCBLP, three different formulations of the problem are taken into account: the first is the original instance, the second is the reformulation in which the valid inequality is considered and added like an upper level constraint, denoted R_1, and the third is the reformulation in

class	(n, m)	(p, q)
1	(10,5)	(5,10)
2	(15,5)	(5,15)
3	(15,10)	(5,20)
4	(20,5)	(5,20)
5	(20,10)	(5,25)

Table 3.7. Classes of small size problems

which the valid inequality is treated like a lower level constraint, denoted R_2. As previously explained, the three formulations were solved using the method proposed by Fortuny-Amat and McCarl. The computational environment is made of a Pentium Core 2 Duo with a 2 GHz processor and 1 GB RAM. The AMPL language and the solver CPLEX 12.3 were used to solve each mathematical formulation. Time limit was fixed to 600 seconds.

The test bed used was randomly generated. We defined two different set of instances, small size and medium size problems. The test problems were generated choosing randomly each parameter in the range $[-50, 50]$.

The small size set is divided into 5 different classes of increasing size problems, each one comprised of 10 instances, combining the number of upper and lower level variables and the number of upper and lower constraints, for a total of 50 instances solved (see Table 3.7).

All these instances were solved to the optimum. The comparison was made taking into account the following features:

– *iter*, the number of MIP simplex iterations

– *CPU*, the CPU time in seconds spent to solve an instance.

We indicated with *original* the DCBLP instance without valid inequality and with R_1 and R_2 the two formulations above described. The column *cut* indicates the computational time necessary to solve problem BLP_{min}^{max} and compute the valid inequality. In Table 3.8 all the results presented are the average values for each class.

The first relevant result is that the computational time required to compute the valid inequality can be considered negligible compared to the time needed to solve the *original* formulation. This means that the time required to compute the cut does not effect the total time, hence it is always possible to compute the cut and include it in the formulation of a given DCBLP instance without a negative effect on the computational effort. Another remarkable result is that the two formulations which include the valid inequality, outperform the original one in all the classes of instances with respect to MIP simplex iterations and CPU time. Except for class 2, the impact of the valid inequality is remarkable in terms of

class	*original*		*cut*	R_1		R_2	
	iter ($\times 10^3$)	*CPU* (sec)	*CPU* (sec)	*iter* ($\times 10^3$)	*CPU* (sec)	*iter* ($\times 10^3$)	*CPU* (sec)
1	210	8.15	0.06	**5**	0.44	**5**	**0.40**
2	114	4.26	0.04	94	3.78	**75**	**3.51**
3	915	33.29	0.12	**618**	**19.19**	631	23.84
4	1206	72.81	0.08	565	45.85	**486**	**40.11**
5	2464	77.67	0.12	1604	**40.30**	**1504**	52.04
average	982	39.24	0.08	577	25.05	540	23.98

Table 3.8. Computational results of the three formulations

class	(n, m)	(p, q)
1	(30,10)	(15,15)
2	(30,15)	(10,20)
3	(40,15)	(15,40)
4	(50,10)	(15,20)
5	(50,15)	(15,25)

Table 3.9. Classes of medium size problems

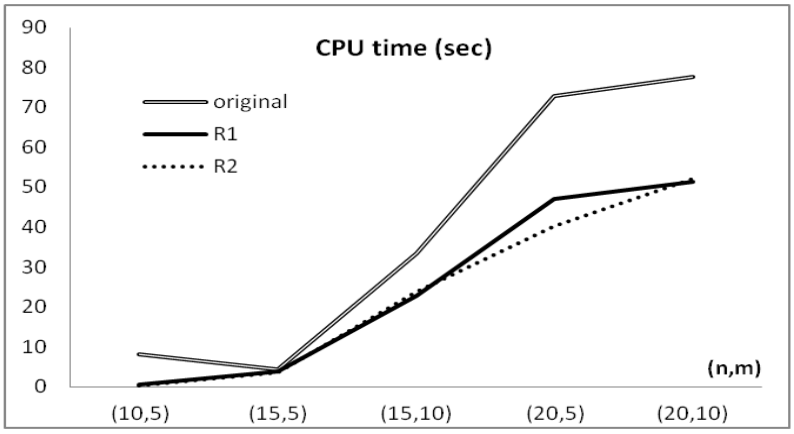
effectiveness: the number of iterations reduction ranges from 17% to 98%, and the CPU time reduction ranges from 11% to 95%. The behaviour of the two formulations is slightly different, and there is not one which clearly dominates the other. Notwithstanding, comparing the average values, formulation R_2 seems to perform slightly better than R_1, suggesting that may be preferable to consider the valid inequality like a lower level constraint. Indeed, the test bed is still too limited and the difference too small, for considering the second formulation definitely better than the first. In general, both the interpretation of the valid cut, as an upper and as lower level constraint, can be implemented obtaining a reduction in the size of the feasible set with a consequent reduction of iterations, thus computational time as shown in Figure 3.3.

A similar analysis was made for the second set of instances of medium size. This set, like the first one, is comprised of 50 instances, divided into 5 different classes of increasing size (see Table 3.9).

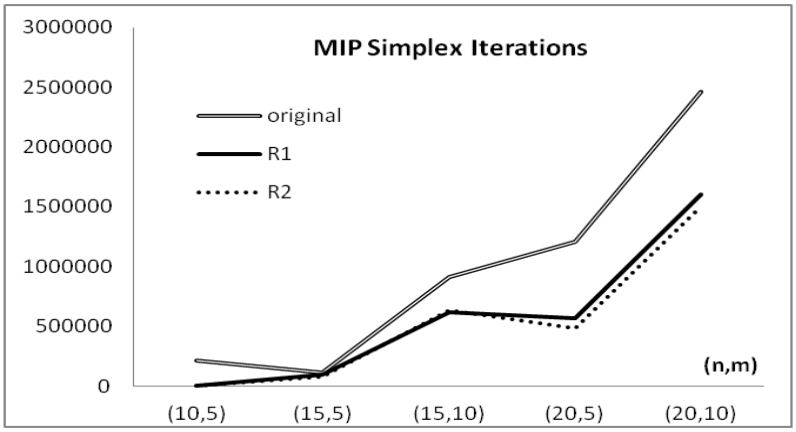
In this second computational analysis, we did not compare the three formulations in terms of CPU time, since the time limit of 600 seconds was reached for all the instances solved but one (an instance of class 2 for formulation R_1). Conversely, we introduced other two indicators:

– Δ, the percentage difference between the solution found by solving *original* and R_1 or R_2;

– *n_inst*, the number of instances for which a feasible solution was found within the time limit.



(a) Computational time



(b) Number of iterations

Figure 3.3. Computational comparison for small size problems

All the average results are reported in Table 3.10.

Similarly to the first set of tests, the CPU time required to compute the valid inequality is negligible and does not effect the general performance of the solution method used. The formulation R_2 is the only one for which in all the instances a feasible solutions is computed within the time limit. Moreover, the quality of solution found is slightly better than the solution computed solving the *original* formulation, especially for classes 2 and 3. A similar result can be observed for the formulation R_1. The NP-hardness of the problem does not allow to make a deeper analysis of the three formulations and imposing a time limit seems to damp the main differences among their computational behaviour. Notwithstanding, the results of formulation R_2 may show a promising impact of the valid inequality both in terms of solution quality and number of iterations, see Figure 3.4.

Finally, in Tables 3.11 and 3.12 we listed in detail the results for all the instances solved.

class	*original*		*cut*	R_1			R_2		
	iter ($\times 10^5$)	*n_inst*	*CPU* (sec)	*iter* ($\times 10^5$)	Δ (%)	*n_inst*	*iter* ($\times 10^5$)	Δ (%)	*n_inst*
1	147	10	0.20	137	-0.21	10	**134**	**-0.28**	10
2	161	9	0.21	**155**	-1.54	10	157	**-1.56**	10
3	142	8	0.22	136	-0.70	9	**134**	**-1.85**	10
4	133	10	0.15	**128**	**-1.05**	10	129	-0.98	10
5	157	10	0.38	156	**-0.28**	10	**154**	-0.27	10
average	148	-	0.23	142	-0.76	-	142	-0.99	-

Table 3.10. Computational results of the three formulations

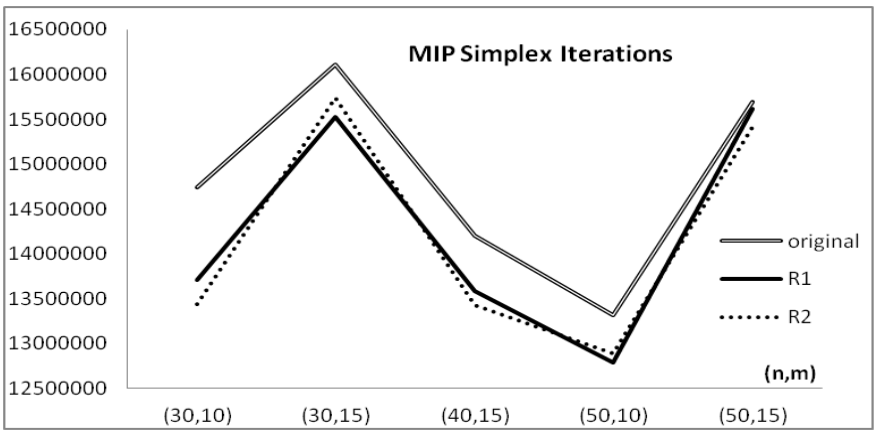


Figure 3.4. Computational comparison of medium size problems

3.2.5 Conclusions

We proposed a valid inequality for a generic DCBLP. The basic idea to compute this valid inequality is to relax the DCBLP, analyze the geometry of its inducible region, that has well known properties, and then obtain information on the bilevel-feasible solutions of the DCBLP. By solving an auxiliary bilevel linear problem it is possible to derive a lower bound on the value of the follower's objective function and then reformulate the original problem taking into account this bound. The rationale behind this approach is to eliminate a subset of solutions that can not be bilevel–feasible exploiting the polyhedral properties of bilevel linear problems.

The proposed inequality is valid both for a BLP and a DCBLP. Notwithstanding, its computation requires to solve another BLP. For this reason it is not convenient to use it for solving a BLP, but it has been proved to be extremely effective if used to solve a DCBLP of small and medium size.

Computational results show that the application of this valid inequality to small size problems not only reduces the number of iterations, but also produces a remarkable computational time reduction of 42% on average. Moreover, the computational effort necessary to compute the inequality (the solution of an auxiliary BLP) is entirely negligible and does not effect the overall performance. These results are obtained if the valid inequality is treated like both an upper and a lower level constraint. Encouraging results can be noticed also for medium size problems solved within a time limit: in this case the valid inequality has a positive impact on the quality of the solution and on the reduction of iterations especially when it is treated as a lower level constraint. Future work may be focused on one side to extend the computational analysis to big size problems in order to make a more comprehensive comparison between the two interpretations of the valid inequality like a leader's or a follower's constraint, and, on the other side, to develop new branch and cut approaches which use the proposed inequalities and exploit the polyhedral properties of bilevel linear problems.

instance	*original*		*cut*	R_1		R_2	
	iter ($\times 10^3$)	*CPU* (sec)	*CPU* (sec)	*iter* ($\times 10^3$)	*CPU* (sec)	*iter* ($\times 10^3$)	*CPU* (sec)
10,5,5,10_1	2	0.16	0.03	2	0.13	2	0.14
10,5,5,10_2	3	0.30	0.06	3	0.44	4	0.42
10,5,5,10_3	11	0.81	0.06	12	0.94	5	0.41
10,5,5,10_4	2	0.26	0.05	2	0.30	1	0.19
10,5,5,10_5	19	0.97	0.09	19	0.89	27	1.11
10,5,5,10_6	2052	78.00	0.03	1	0.20	1	0.19
10,5,5,10_7	2	0.14	0.09	2	0.19	1	0.22
10,5,5,10_8	1	0.22	0.05	1	0.19	1	0.13
10,5,5,10_9	4	0.36	0.08	3	0.25	3	0.28
10,5,5,10_10	2	0.28	0.06	2	0.25	2	0.27
15,5,5,15_1	54	4.49	0.03	54	4.42	42	3.26
15,5,5,15_2	40	1.16	0.05	40	1.19	52	1.47
15,5,5,15_3	16	1.01	0.03	16	1.00	21	1.23
15,5,5,15_4	105	4.17	0.03	105	4.18	117	4.76
15,5,5,15_5	5	0.28	0.03	5	0.28	6	0.32
15,5,5,15_6	48	1.70	0.05	48	1.29	8	0.48
15,5,5,15_7	426	14.24	0.03	234	9.39	275	12.59
15,5,5,15_8	26	1.05	0.03	22	1.01	22	1.00
15,5,5,15_9	375	11.98	0.06	375	12.03	164	6.90
15,5,5,15_10	41	2.54	0.05	41	2.59	41	2.67
15,10,5,20_1	620	23.98	0.14	453	17.74	304	12.25
15,10,5,20_2	2406	92.77	0.11	1212	45.61	1610	69.75
15,10,5,20_3	262	13.12	0.09	262	12.98	144	8.47
15,10,5,20_4	766	16.85	0.11	516	11.83	585	13.15
15,10,5,20_5	94	4.93	0.09	94	4.76	42	2.45
15,10,5,20_6	407	11.97	0.16	442	12.50	368	9.81
15,10,5,20_7	2882	119.33	0.16	1584	72.68	2064	82.90
15,10,5,20_8	231	6.91	0.09	231	6.93	84	2.82
15,10,5,20_9	643	16.85	0.09	643	16.69	226	7.49
15,10,5,20_10	836	26.18	0.11	741	23.88	880	28.31
20,5,5,20_1	252	11.06	0.05	76	5.30	76	5.33
20,5,5,20_2	3109	287.54	0.08	2659	250.61	2542	259.48
20,5,5,20_3	46	3.81	0.05	46	3.78	46	3.99
20,5,5,20_4	388	18.80	0.06	388	18.56	388	18.47
20,5,5,20_5	71	5.24	0.11	71	5.10	56	3.64
20,5,5,20_6	4677	202.19	0.14	1447	70.11	455	29.56
20,5,5,20_7	119	8.19	0.08	123	8.28	107	7.96
20,5,5,20_8	73	3.12	0.13	37	1.87	43	2.22
20,5,5,20_9	3052	169.23	0.05	524	86.89	943	54.91
20,5,5,20_10	275	18.89	0.06	275	18.79	202	14.74
20,10,5,25_1	569	25.52	0.11	413	15.68	653	28.25

instance	*original*		*cut*	R_1		R_2	
	iter ($\times 10^3$)	*CPU* (sec)	*CPU* (sec)	*iter* ($\times 10^3$)	*CPU* (sec)	*iter* ($\times 10^3$)	*CPU* (sec)
20,10,5,25_2	4791	165.20	0.08	2820	97.64	1164	61.12
20,10,5,25_3	374	18.19	0.13	374	12.09	95	3.84
20,10,5,25_4	2199	85.54	0.08	3075	107.31	2617	116.08
20,10,5,25_5	3272	88.61	0.06	1335	43.34	1605	55.60
20,10,5,25_6	1630	72.17	0.53	970	35.51	1426	47.08
20,10,5,25_7	426	11.19	0.08	196	6.12	235	7.53
20,10,5,25_8	275	12.64	0.05	114	6.19	580	17.41
20,10,5,25_9	7552	187.65	0.06	2899	73.88	4536	108.98
20,10,5,25_10	3550	110.01	0.06	3838	114.75	2133	73.24

Table 3.11. Computational results in details for each small size instance solved

instance	*original*		*cut*	R_1		R_2	
	iter ($\times 10^5$)	*solution*	*CPU* (sec)	*iter* ($\times 10^5$)	Δ (%)	*iter* ($\times 10^5$)	Δ (%)
30,10,15,15_1	243	-3765.91	0.16	152	-0.26	141	-0.14
30,10,15,15_2	145	-2826.13	0.08	134	0.23	147	0.31
30,10,15,15_3	147	-1466.74	0.09	150	-2.26	147	-2.50
30,10,15,15_4	99	-2447.78	0.06	95	0.00	82	-0.16
30,10,15,15_5	148	-3141.03	0.09	167	0.55	145	0.03
30,10,15,15_6	137	-2213.47	1.12	122	-0.27	100	-0.33
30,10,15,15_7	124	-4558.59	0.11	126	0.00	165	-0.01
30,10,15,15_8	140	-3933.31	0.06	140	0.00	148	0.23
30,10,15,15_9	176	-2472.47	0.11	171	0.00	158	-0.01
30,10,15,15_10	116	-2376.17	0.06	114	-0.03	112	-0.22
30,15,10,20_1	200	-2890.05	0.28	159	0.72	158	0.78
30,15,10,20_2	155	-1851.53	0.20	157	-15.72	131	-15.72
30,15,10,20_3	145	-2628.96	0.13	176	0.98	160	0.66
30,15,10,20_4	129	-2597.15	0.09	115	-0.33	114	-0.33
30,15,10,20_5	204	-4050.66	0.19	201	0.00	211	0.28
30,15,10,20_6	186	-5170.45	0.52	88	-0.13	175	-0.05
30,15,10,20_7	162	-	0.19	209	-	211	-
30,15,10,20_8	177	-3128.19	0.17	199	0.66	161	0.04
30,15,10,20_9	126	-3098.92	0.05	124	0.00	123	0.09
30,15,10,20_10	127	-3983.52	0.25	124	-0.07	131	0.16
40,15,15,40_1	145	-4246.53	0.11	150	0.35	145	0.44
40,15,15,40_2	151	-3131.70	0.62	137	-3.44	134	-3.96
40,15,15,40_3	152	-3466.01	0.08	152	-0.61	143	0.22
40,15,15,40_4	125	-3232.03	0.13	127	-	125	-3.80
40,15,15,40_5	137	-	0.16	137	-	101	-

instance	*original*		*cut*	R_1		R_2	
	iter ($\times 10^5$)	*solution*	*CPU* (sec)	*iter* ($\times 10^5$)	Δ (%)	*iter* ($\times 10^5$)	Δ (%)
40,15,15,40_6	121	-	0.09	127	-	131	-
40,15,15,40_7	120	-4090.18	0.59	121	0.00	163	-5.90
40,15,15,40_8	157	-3781.86	0.11	149	-0.86	108	-1.59
40,15,15,40_9	147	-4141.10	0.16	146	0.29	126	-0.11
40,15,15,40_10	164	-3422.21	0.17	111	-0.64	166	-0.11
50,10,15,20_1	159	-4954.07	0.19	149	-0.09	176	0.05
50,10,15,20_2	146	-5457.03	0.27	125	-9.43	138	-9.68
50,10,15,20_3	79	-6306.29	0.08	97	0.07	105	0.30
50,10,15,20_4	131	-6679.57	0.09	121	-0.43	122	-0.37
50,10,15,20_5	127	-7106.76	0.31	127	0.00	121	-0.26
50,10,15,20_6	88	-7103.65	0.14	115	0.02	126	-0.01
50,10,15,20_7	149	-5209.96	0.13	150	0.03	115	0.20
50,10,15,20_8	194	-6639.24	0.08	139	-0.62	139	0.31
50,10,15,20_9	125	-4883.42	0.09	123	0.00	123	0.00
50,10,15,20_10	133	-5042.59	0.13	133	0.00	125	-0.30
50,15,15,25_1	144	-6227.10	0.30	148	0.00	145	-0.37
50,15,15,25_2	184	-6907.57	1.30	179	-0.06	173	-0.14
50,15,15,25_3	145	-4607.55	0.09	143	-1.95	139	-1.71
50,15,15,25_4	149	-8101.07	0.42	156	0.34	136	-0.11
50,15,15,25_5	133	-5881.63	0.25	131	0.22	146	0.23
50,15,15,25_6	151	-5404.41	0.34	151	0.03	125	-0.52
50,15,15,25_7	104	-6697.39	0.37	108	0.06	104	1.41
50,15,15,25_8	181	-6971.47	0.23	177	-0.09	177	-0.06
50,15,15,25_9	198	-6908.01	0.38	201	-1.25	203	-1.25
50,15,15,25_10	180	-7900.03	0.09	167	-0.11	193	-0.17

Table 3.12. Computational results in details for each medium size instance solved

Chapter 4

Discrete Bilevel Linear Programming

In this chapter we focus on the study of bilevel linear programming problems in which all the variables are discrete. The main computational complexities are analyzed and two different lines of research are followed: the development of new exact methods and new fast heuristics. Despite an increasing interest of scientific community towards bilevel optimization problems, the class of Discrete Bilevel Linear Problems (DBLPs) has not been been sufficiently investigated in our opinion. In this chapter we describe new theoretical results for DBLP that are used to design two new exact algorithms and two new heuristic algorithms. Some of these results directly stem from similar properties investigated for the discrete–continuous case and are properly readapted for DCBLP. The main challenge originates from the integer nature of the follower problem. For this reason the majority of results available for BLP are not still valid for this class of problems and new solution approaches are pursued.

4.1 Enhanced exact algorithms for DBLP

Let us recall the general formulation of a Discrete Bilevel Linear Problem (DBLP):

$$
\begin{aligned}
(DBLP) \quad & \min_{x,y} F(x,y) = c_1^T x + c_2^T y \\
& \text{s.t.} \quad Cx + Dy \leq e \\
& \qquad x \in \mathbb{Z}_+ \\
& \qquad y \in \operatorname*{argmin}_{y} f(y) = d^T y \\
& \qquad\quad \text{s.t.} \quad Ax + By \leq b \\
& \qquad\qquad y \in \mathbb{Z}_+
\end{aligned}
$$

For the sake of simplicity, in the rest of this chapter we assume there are not upper level constraints. In the literature there are very few references on DBLPs. Some results have been shown by Bard and Moore [24] for the binary case and by Dempe [54] for problems in which the follower's feasible set is not parameterized by x and the follower's objective function is bilinear. Mansi et al. [105] propose an exact method for a DBLP with a follower binary knapsack problem. DeNegre and Ralphs [57] propose a branch and cut scheme for DBLPs based on a simple valid inequality that eliminates an integer solution after checking it is not bilevel–feasible. In the following, this algorithm is used like benchmark for a computational comparison to the exact methods we propose.

In this section we present two novel exact algorithms for the resolution of a generic DBLP. The first algorithm is a cutting plane approach. The second algorithm is based on an existing branch and cut properly modified to exploit some geometric properties of bilevel linear problems. The main goal of this section is to present two new exact algorithms and to compare their computational performances. The new algorithms, that are proved to be very efficient, represent a clear contribution to the state of the art of solution methods for DBLPs.

4.1.1 Preliminaries

In Chapter 2 we showed the relation between the general formulation of a bilevel linear problem and its single level relaxation. For the sake of clearness, let us recall the same definition for the discrete case. The single level relaxation (SLR) of a DBLP without upper level constraints is

$$\begin{aligned} (SLR) \quad & \min_{x,y} F(x,y) = c_1^T x + c_2^T y \\ & \text{s.t.} \quad Ax + By \leq b \\ & \qquad x \in \mathbb{Z}_+ \\ & \qquad y \in \mathbb{Z}_+ \end{aligned}$$

In other words, in SLR the follower's objective function is dropped, thus the reaction set $R_y(x)$ and the inducible region IR are not defined and the problem is solved on the feasible region S. Note that in SLR the variables are required to be integer. Let us define with (x_1, y_1) and (x_2, y_2) the optimal solutions of DBLP and SLR, respectively. Moreover we define with (x_3, y_3) the optimal solution of the continuous relaxation of SLR. We have:

$$F(x_3, y_3) \leq F(x_2, y_2) \leq F(x_1, y_1)$$

Consider the following problem P_1:

$$\begin{aligned} (P_1) \quad & \min_{x,y} F(x,y) = 2x + 7y \\ & \text{s.t.} \quad x \in \mathbb{Z}_+ \\ & \qquad y \in \min_{y} f(y) = -y \\ & \qquad\quad \text{s.t.} \quad 2x - 8y \geq -25 \\ & \qquad\qquad 7x + 10y \leq 60 \\ & \qquad\qquad 2x + y \geq 6 \\ & \qquad\qquad 11x - 4y \leq 31 \\ & \qquad\qquad y \in \mathbb{Z}_+ \end{aligned}$$

In Figure 4.1 all the integer solutions of P_1 are represented. In this case $(x_1, y_1) \equiv A$, $(x_2, y_2) \equiv E$ and $(x_3, y_3) \equiv F$, thus we obtain $F(x_1, y_1) = 25$, $F(x_2, y_2) = 13$, $F(x_3, y_3) \approx 7.3$.

Let us drop the integrality requirements of problem DBLP preserving its bilevel structure and let us denote with (x_4, y_4) the optimal solution of this problem. It is not possible to define any relation between $F(x_1, y_1)$ and $F(x_4, y_4)$ (see Moore and Bard [114] for more details). In order to clarify this issue, let us consider the linear relaxation of problem P_1, denoted as P_2:

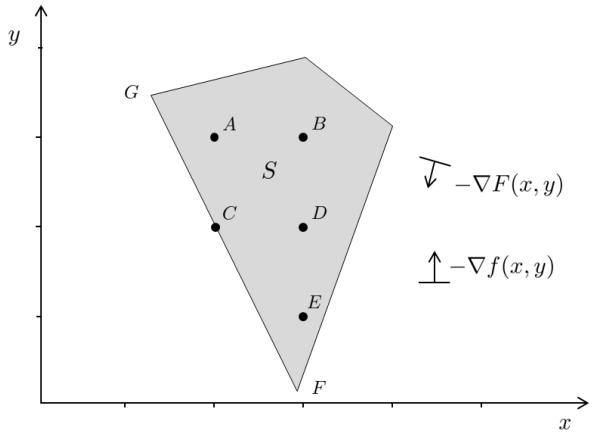


Figure 4.1. Set of integer solutions and extreme points of S

$$
\begin{aligned}
(P_2) \quad & \min_{x,y} F(x,y) = 2x + 7y \\
& x \geq 0 \\
& y \in \underset{y}{\text{argmin}}\ f(y) = -y \\
& \quad \text{s.t.} \quad 2x - 8y \geq -25 \\
& \qquad\quad 7x + 10y \leq 60 \\
& \qquad\quad 2x + y \geq 6 \\
& \qquad\quad 11x - 4y \leq 31 \\
& \qquad\quad y \geq 0
\end{aligned}
$$

In Figure 4.2 the two inducible regions are shown. Let us call them IR_{P_1} and IR_{P_2} for problems P_1 and P_2 respectively. IR_{P_1} is represented by the bold line (note that it is comprised of two support hyperplanes of the feasible region) and IR_{P_2} is comprised of the black points (while the white points are the integer not bilevel–feasible solutions).
It is easy to see that $IR_{P_1} \nsubseteq IR_{P_2}$: therefore, it is not surprising that the optimal solution of the relaxed problem P_2 is $(x_4, y_4) \equiv G$ with $F(x_4, y_4) \approx 26.7 > F(x_1, y_1)$.

Summing up, the linear relaxation of a DBLP does not provide a valid lower bound in general.

4.1.2 A cutting plane method

We now introduce the first exact algorithm based on a cutting plane approach. In the previous section we showed that a valid lower bound for DBLP can be computed solving SLR which is a single level discrete problem. The optimal solution of SLR, denoted as $(\bar{x}, \bar{y})$, is integer, but may not be bilevel–feasible: it means that in the follower's feasible region $\Omega_y(\bar{x})$ there may exist a solution y^* such that $f(y^*) < f(\bar{y})$, that is $\bar{y}$ is not the best response for the follower. The rationale behind our algorithm is to solve SLR and compute a lower bound $F(\bar{x}, \bar{y})$. If the solution is bilevel–feasible it provides also an upper bound, hence, it is optimal; otherwise, there exists a bilevel–feasible solution $(\bar{x}, y^*)$ and we introduce a cut to eliminate all the solutions which are not bilevel–feasible at $x = \bar{x}$ and then iterate. The valid inequality is the following:

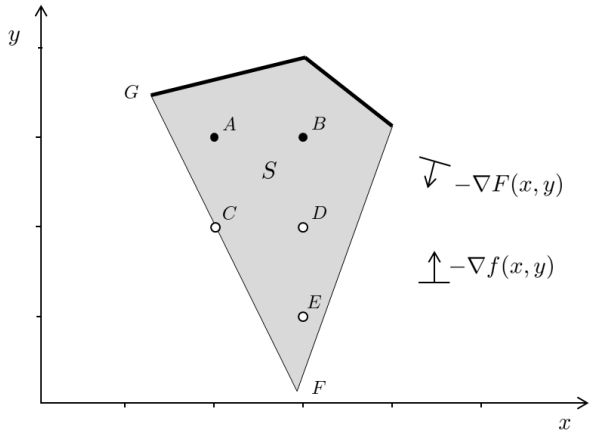


Figure 4.2. Comparison between two inducible regions

$$f(y) \leq f(y^*) + L \cdot |x - \bar{x}|$$

where L is a parameter greater than the maximum value that function $f(y)$ can achieve and that can be easily computed. Basically, the idea is that the cut works only when $x = \bar{x}$ and it is inactive otherwise. This formulation of the cut is not linear but can be reformulated as follows:

$$f(y) \leq f(y^*) + L \cdot z(\bar{x})$$

where $z(\bar{x})$ is defined as the optimal solution of this auxiliary problem $P_z(\bar{x})$

$$\begin{aligned} (P_z(\bar{x})) \quad & \min_z z \\ & \text{s.t.} \quad z \geq x - \bar{x} \\ & \qquad z \geq \bar{x} - x \end{aligned}$$

Thus, every time a not bilevel–feasible solution is found, we introduce in the SLR a follower problem comprised of one free auxiliary variable z and $2n$ constraints. The main advantage is that the resulting proble can be treated and solved like a bilevel linear problem with continuous lower level variables. Indeed, as pointed out in the literature, the main difficulty of a DBLP is the discrete nature of the follower problem. In this cutting plane approach the follower problem $P_z(\bar{x})$ is a continuous problem and incorporating it within the SLR the resulting problem is a bilevel problem in which all the discrete variables are under control of the leader.

Given a not bilevel–feasible solution $(\bar{x}, \bar{y})$ and its corresponding bilevel–feasible one $(\bar{x}, y^*)$, the resulting SLR formulation after the cut generation is:

$$
\begin{aligned}
\min_{x,y}\ & F(x,y) = c_1^T x + c_2^T y \\
\text{s.t.}\ & x \in \mathbb{Z}_+ \\
& y \in \mathbb{Z}_+ \\
& f(y) = d^T y \\
& f(y) \le f(y^*) + L \cdot z(\bar{x}) \\
& z(\bar{x}) \in \underset{z}{\text{argmin}}\ z \\
& \quad \text{s.t.}\ \ z \ge x - \bar{x} \\
& \qquad\ \ z \ge \bar{x} - x
\end{aligned}
$$

Here is the detail of the cutting plane algorithm:

Step 1 Solve problem SLR and compute a lower bound $LB = F(\bar{x}, \bar{y})$.

Step 2 (Bilevel-feasibility check) Solve the follower problem. If the solution is bilevel–feasible, $(\bar{x}, \bar{y})$ is optimal then STOP, else go to Step 3.

Step 3 (Cutting plane) Modify problem SLR with an additional follower problem $P_z(\bar{x})$ and go to Step 1.

Note that this algorithm can be used even if the set of leader's constraints is non empty, that is $p \neq 0$, or if we remove the semi-cooperative assumption between the two decision makers. In this case, every time a solution is checked not to be stable through a stability check, it can be removed with the same cutting procedure. A stability test, as showed in Chapter 1, is described in Bianco et al. [35]: assuming that solution $(\bar{x}, y^*)$ is bilevel–feasible, by solving the following

$$
\begin{aligned}
\max_{y}\ & F(\bar{x}, y) \\
\text{s.t.}\ & (\bar{x}, y) \in S \\
& f(y) \le f(y^*)
\end{aligned}
$$

if the solution is non stable a new bilevel–feasible solution is found which is worse for the leader.

Similarly to the previous chapter, the method we use for solving bilevel linear problem with continuous lower level variables is the one proposed by Fortuny-Amat and McCarl. In this case, the follower $P_z(\bar{x})$ is replaced by the following set of constraints

$$
\begin{aligned}
& z - x + \bar{x} = t && 1 - \mu \le M \cdot \beta \\
& z - \bar{x} + x = s && s \le M \cdot (1 - \beta) \\
& \mu \le M \cdot \alpha && \mu, t, s \in \mathbb{R}^n_+ \\
& t \le M \cdot (1 - \alpha) && \alpha, \beta \in \mathbb{B}^n
\end{aligned}
$$

This reformulation can not be used for DBLP, since the KKT conditions are not necessary and sufficient optimality conditions for the follower problem. This method is widely used to solve common bilevel linear problems as it is easily implementable and provides a mixed-integer single level problem. The main drawback is that the complexity of the MIP grows proportionally to the size of the follower problem and for big size problems many approximation errors, due to the presence of big-M, may occur. In our algorithm, the follower problem $P_z(\bar{x})$ is of fixed size (one variable and $2n$ constraints), but a new follower problem is added in every iteration. In order to avoid a large number of follower problems that are difficult to handle, we propose a slight modification of the cutting plane algorithm.

4.1.3 Modified cutting plane

In this modified version we try to reduce the number of cutting planes added to the SLR. The basic idea stems from an observation. Let us assume that a solution $(\bar{x}, \bar{y})$ is found which is not bilevel–feasible. According to the algorithm, we introduce the follower problem $P_z(\bar{x})$ and cut off the solution.

Let (x', y') be the new not bilevel–feasible solution providing a new lower bound. If solution (x', y') is far from $(\bar{x}, \bar{y})$ in the feasible region S, it is reasonable to suppose that the descent direction of function $F(x, y)$ may lead to solutions far from $(\bar{x}, \bar{y})$. It implies that it may be possible to drop the cutting plane we introduced, i.e. remove the follower problems $P_z(\bar{x})$, and replace it with the inequality $F(x, y) \geq F(x', y')$, which is violated by $(\bar{x}, \bar{y})$. Notwithstanding, if in a subsequent iteration a new solution $(\bar{x}, \hat{y})$ is found, it means that the algorithm computes another not bilevel–feasible solution for $x = \bar{x}$: for avoiding this cycling effect, in this case the follower problem $P_z(\bar{x})$ is reintroduced and it is no longer dropped. This modified version, on the one hand, may lead to an increased number of iterations, but on the other hand, reduces the number of unnecessary follower problems which can be extremely effective when large size instances are solved.

4.1.4 An example

Let us recall the previous example

$$
\begin{aligned}
\min_{x,y} \; & F(x, y) = 2x + 7y \\
\text{s.t.} \quad & x \in \mathbb{Z}_+ \\
& y \in \min_{y} \; f(y) = -y \\
& \quad \text{s.t.} \quad 2x - 8y \geq -25 \\
& \qquad\quad 7x + 10y \leq 60 \\
& \qquad\quad 2x + y \geq 6 \\
& \qquad\quad 11x - 4y \leq 31 \\
& \qquad\quad y \in \mathbb{Z}_+
\end{aligned}
$$

The set of feasible solutions is depicted in Figure 4.1. Solutions A and B are bilevel–feasible, while C, D and E are not. Applying the first version of the algorithm, the initial solution is $E = (3, 1)$. E is not bilevel–feasible since the optimal follower's reaction for $x = 3$ is solution $B = (3, 3)$ with $f(3) = -3$. The cut $f(y) \leq -3 + L \cdot |x - 3|$ is applied, thus if $x = 3$ the only feasible solution is B. The second solution is $C = (2, 2)$ and it is not bilevel–feasible too. The algorithm computes the optimal solution $A = (2, 3)$ at the third iteration. Solving the same instance through the modified version, after solution C, the cut $f(y) \leq -3 + L \cdot |x - 3|$ is removed and it is replaced by the inequality $F(x, y) \geq 18$, where $F(C) = 18$. At the third iteration the

algorithm computes solution $D = (3, 2)$ that would have not been found if the cut had not been dropped. To avoid cycling on already found solutions, cut $f(y) \leq -3 + L \cdot |x - 3|$ is restored and the optimal solution A is computed at the fourth iteration.

Let us modify leader's objective function as $F(x, y) = 8x + 7y$. Now the first solution is C, then solution E is found with both algorithms. In this case, if the modified version is used, the valid inequality added to make solution C infeasible is dropped and the optimal solution A is found at the second iteration with the advantage of solving a less constrained problem.

4.1.5 A branch and cut algorithm

One of the main differences between DBLPs and integer linear programming problems is that a solution (x, y) is required to be in the feasible set S, but also to be bilevel–feasible, that is $y \in R_y(x)$. It follows that an integer solution does not necessarily provide a valid upper bound unless it belongs to the inducible region IR.

As stated above, SLR represents a valid relaxation of DBLP. If a branch and bound approach is used to solve SLR, every time an integer solution (x, y) is computed at a given node problem, $F(x, y)$ is a lower bound. The node can be pruned if the solution provides also a valid upper bound, i.e. is bilevel–feasible. In the literature a method proposed by DeNegre and Ralphs features the application of a valid inequality to eliminate a solution $(x, y) \in S$ such that $y \neq R_y(x)$. This method, that will be described more in detail in the following section, is the only existing branch and cut approach to solve a generic DBLP to our knowledge and it will be used as a benchmark in the computational analysis reported below.

The main idea of our new branch and cut method originates from the geometrical properties of bilevel linear problems highlighted in Chapter 3: IR is piecewise linear and is comprised of supporting hyperplanes of S. As a consequence, IR is a non convex set, but it is a small subset of S. Our idea is to exploit this property to reformulate S defining a smaller size feasible region S' and solve DBLP on S'. This idea is an adaptation of the valid inequality provided for DCBLP in Chapter 3. Recall the max-min auxiliary problem, denoted as BLP^{max}_{min}:

$$
\begin{aligned}
(BLP^{max}_{min}) \quad & \max_{x,y} f(y) = d^T y \\
& \text{s.t.} \quad x \in \mathbb{R}^n_+ \\
& \qquad y \in \operatorname*{argmin}_y f(y) = d^T y \\
& \qquad\quad \text{s.t.} \quad Ax + By \leq b \\
& \qquad\qquad y \in \mathbb{R}^m_+
\end{aligned}
$$

Let $(\hat{x}, \hat{y})$ be the optimal solution. We have already proved that, for a BLP and a DCBLP $f(\hat{y})$ is an upper bound on the value of the follower's objective function. Let us consider the following inequality

$$f(y) \leq \lceil f(\hat{y}) \rceil$$

This is a valid inequality for BLP and DCBLP. In the previous example, the optimal solution of BLP^{max}_{min} is $(\hat{x}, \hat{y}) \approx (4, 3.2)$. Note that adding the inequality to the original formulation of S, the new feasible set, denoted as S', comprises only the two bilevel–feasible solutions A and B.

Unfortunately the validity of this inequality does not always hold for DBLP and it may happen that S' does not contain the optimal solution. In this section we show how to readapt this inequality in order to provide a new

exact method for DBLP. In the following section we explain from a theoretical point of view why this is not a valid inequality for DBLP, we use it to design a heuristic approach and present two special cases in which the validity for DBLP holds.

Preliminary tests show that the new feasible set S' is very likely to contain the optimal solution. For this reason, we propose to use this inequality as a branching rule. The original feasible set S is splitted in two subset, S' and S'' adding inequalities $f(y) \leq \lceil f(\hat{y}) \rceil$ and $f(y) \geq \lceil f(\hat{y}) \rceil + 1$ respectively. Thus from the original problem two subproblems DBLP$'$ and DBLP$''$ are generated.

The general framework of our branch and cut algorithm is described below.

Step 1 Solve problem BLP_{min}^{max} and compute the optimal solution $(\hat{x}, \hat{y})$. Define sets S' and S'' as:

$$S' = S \cap \{f(y) \leq \lceil f(\hat{y}) \rceil\}$$
$$S'' = S \cap \{f(y) \geq \lceil f(\hat{y}) \rceil + 1\}$$

and problems DBLP$'$ and DBLP$''$, then go to Step 2.

Step 2 Solve problem DBLP$'$ and compute an incumbent solution UB'. Go to Step 3.

Step 3 Solve SLR$''$, that is the single level relaxation of DBLP$''$, to compute a lower bound LB''. If $LB'' \geq UB'$ close DBLP$''$ and STOP, else got to Step 4.

Step 4 Solve problem DBLP$''$. Every time a new LB'' is computed check if DBLP$''$ can be closed, otherwise continue.

Problems DBLP$'$ and DBLP$''$ are solved by means of the above mentioned branch and cut algorithm existing in the literature that is described more in detail in the following sections. The crucial step of our algorithm is Step 3: if the relaxation of problem DBLP$''$ provides a lower bound worse than the incumbent, this certificates that the optimal solution has already been found in S' and the algorithm halts.

4.1.6 Hybrid branch and cut

Even if the optimal solution is contained in S', some preliminary tests show that many lower bounds have to be computed in Step 4 before DBLP$''$ can be closed and this represents a waste of time in terms of efficiency. Note that a lower bound is computed solving the relaxed version of SLR which is a linear programming problem. With the goal of reducing the computational time in Step 4, we propose to solve DBLP$''$ using the first version of the cutting plane algorithm described in Section 4.1.2. In this case, in order to compute a lower bound, it is necessary to solve SLR which is an integer linear programming problem. Thus, there is a trade off between the two methods in terms of number of iterations and computational time per iteration.

Summing up, in this hybrid version of the algorithm, DBLP$'$ is solved by the branch and cut algorithm existing in the literature and DBLP$''$ is solved by our cutting plane algorithm.

Let us consider the same example above mentioned. The feasible sets S' and S'' defined after Step 1 are depicted in Figure 4.3.

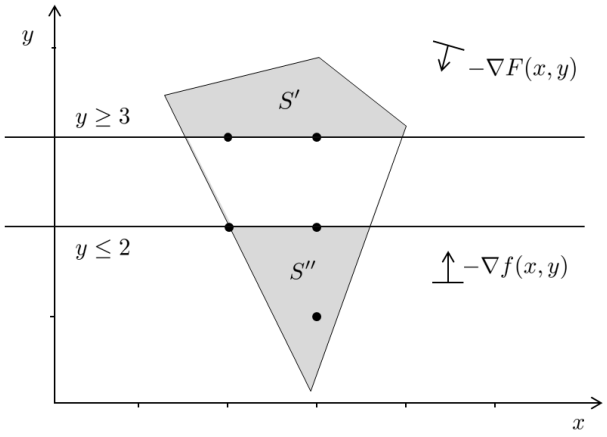


Figure 4.3. Set S splitted into S' and S''

The optimal solution is in S' and $UB' = 25$. Solving a relaxation of DBLP$''$ we obtain $LB'' \approx 7.3 < 25$ thus the algorithm can not stop. If we apply the first version of the algorithm at least 8 iterations have to be done before certificating that S'' has not bilevel–feasible solutions. The same result is obtained applying the cutting plane algorithm of Section 3 in only 3 iterations.

4.1.7 Computational analysis

In this section, we report and analyze the computational performance of the methods above proposed: the cutting plane algorithm (CP) and its modified version (MCP), the branch and cut algorithm (BC) and its hybrid version (HBC). The branch and cut method developed by DeNegre and Ralphs (DR) is used like a benchmark as it represents one of the best performing algorithm existing in the literature to solve a generic instance of DBLP.

The test bed used for the our computational results is formed by randomly generated problems of different size following the same rationale used for the computational analysis described in Chapter 3. As the complexity strictly depends on the number of upper and lower level variables, we set $n \in \{5, 10, 15\}$ and $m \in \{5, 10, 15\}$ and we defined 7 different problems combining the value of n and m and setting the number of constraints $q = n + m$. 10 instances were generated for each problem choosing randomly each parameter in the range $[-50, 50]$. All the algorithms were implemented in the C language and the test was performed on a PC Pentium Core 2 Duo with a 2 GHz processor and 1 GB RAM. Solver CPLEX 12.3 was used within the algorithms to solve test problems.

In Table 4.1, all the notations used in this section are summarized.

4.1.8 Cutting plane algorithms

We compared CP and MCP to the benchmark algorithm DR. The results are listed in Table 4.2 and have to be considered as average values out of 10 instances. The best results are reported in bold.

The comparison was made with regards to number of iterations and computational times. The number of

DR	benchmark branch and cut algorithm proposed by DeNegre and Ralphs
BC	branch and cut algorithm described in Section 4.1.5
HBC	hybrid branch and cut algorithm described in Section 4.1.6
CP	cutting plane algorithm described in Section 4.1.2
MCP	modified cutting plane algorithm described in Section 4.1.3
iter	total number of iterations
ott/iter	percentage ratio between the iteration in which the optimum is found and iter
CPU	CPU time spent in seconds

Table 4.1. Notations

(n, m)	DR		CP		MCP	
	iter	*CPU*	*iter*	*CPU*	*iter*	*CPU*
(10,5)	2007	218.03	**25**	80.90	35	**39.91**
(5,10)	1568	284.32	**17**	**23.67**	27	30.14
(10,10)	2264	362.99	**23**	40.98	31	**38.95**
(15,5)	1704	181.28	**21**	31.58	23	**28.04**
(5,15)	2399	267.33	**10**	**14.42**	16	18.21
(15,10)	4190	452.09	**29**	**60.06**	43	71.96
(10,15)	2640	314.87	**16**	**25.43**	25	28.90
average	2396	297.27	**20**	39.58	28	**36.45**

Table 4.2. Comparison of CP, MCP and the benchmark algorithm DR

iterations of CP and MCP are two orders of magnitude smaller than the iterations of DR, while the CPU time is one order of magnitude smaller. Both CP and MCP are extremely faster than DR: CP solves all instances in an average time smaller than 45 seconds but two cases (80.90 and 60.06 seconds) and MCP performs even better with all instances but one (71.96 seconds) solved within 40 seconds. As expected the number of iterations made by MCP is greater than the iterations of CP, but this is counterbalanced by a computational time that is smaller for problems $(10, 5)$, $(10, 10)$ and $(15, 5)$ and almost comparable for problems $(5, 15)$ and $(10, 15)$. Even if the average value of CPU time seems to show that MCP should be preferred to CP, there is not a clear best performing result of one algorithm compared to the other. Note that iterations and CPU time of CP and MCP always decrease as n and m are swapped and this happens mainly because the follower problems we add in the cutting procedure have a size that strictly depends of n. This phenomenon does not happen for DR.

4.1.9 Branch and cut algorithms

Branch and cut algorithms were compared to each other and to the benchmark algorithm in terms of iterations and CPU time. In HBC we set a time limit of 120 second for solving DBLP$''$. Moreover we introduce an additional indicator, i.e. the ratio between the instance in which the optimal solution is computed and the total number of iterations. The latter is an efficiency indicator for an algorithm: a low percentage ratio means that an optimal solution is found early, but is not certified as optimal until a large number of iterations is performed, on the contrary, a high percentage ratio implies that the algorithm does not waste too much computational time and it is more efficient. This indicator is particularly useful to compare BC and HBC. As we pointed out in the previous section, the main disadvantage of BC occurs if the optimal solution is contained in DBLP$'$, but it is necessary to solve DBLP$''$, too. Speeding up the resolution of the latter problem with HBC, we may expect an increased percentage ratio. Finally note that this indicator is meaningless for the CP and MCP since they stop once an optimal solution is found, hence the percentage ratio is always 100%. All the average results are reported in Table 4.3.

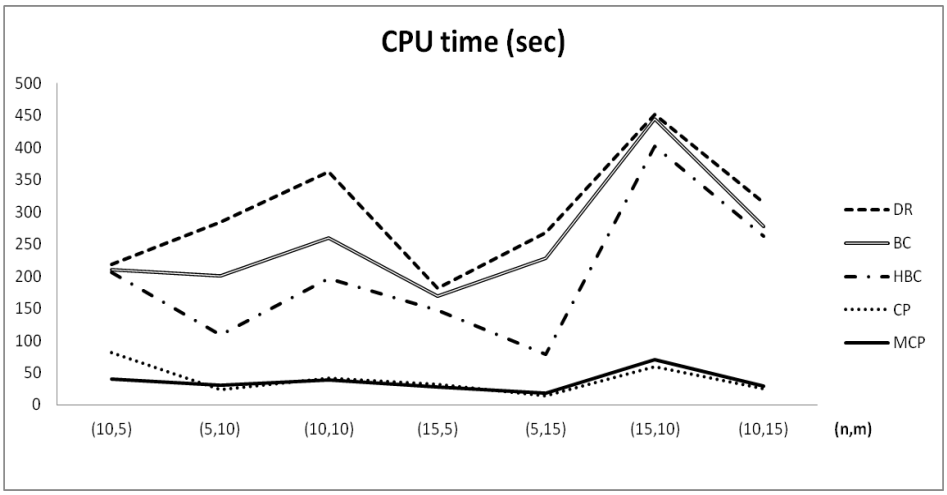
(n, m)	DR			BC			HBC		
	iter	*ott/iter*	*CPU*	*iter*	*ott/iter*	*CPU*	*iter*	*ott/iter*	*CPU*
(10,5)	2007	**83.76%**	218.03	1904	70.70%	210.27	**1739**	79.72%	**205.59**
(5,10)	1568	61.91%	284.32	1742	43.22%	200.63	**1114**	58.81%	**109.04**
(10,10)	2264	73.61%	362.99	2263	66.60%	259.56	**1524**	**81.67%**	**196.87**
(15,5)	1704	**69.06%**	181.28	1541	49.43%	169.69	**1276**	59.82%	**147.14**
(5,15)	2399	71.06%	267.33	2045	44.50%	227.98	**1684**	50.65%	**78.51**
(15,10)	4190	**83.59%**	452.09	3957	62.51%	445.00	**3504**	63.62%	**402.17**
(10,15)	2640	78.89%	314.87	2493	76.22%	277.47	**2286**	**79.23%**	**262.58**
average	2396	**74.55%**	297.27	2278	59.03%	255.80	**1875**	67.65%	**200.27**

Table 4.3. Comparison of BC, HBC and the benchmark algorithm DR

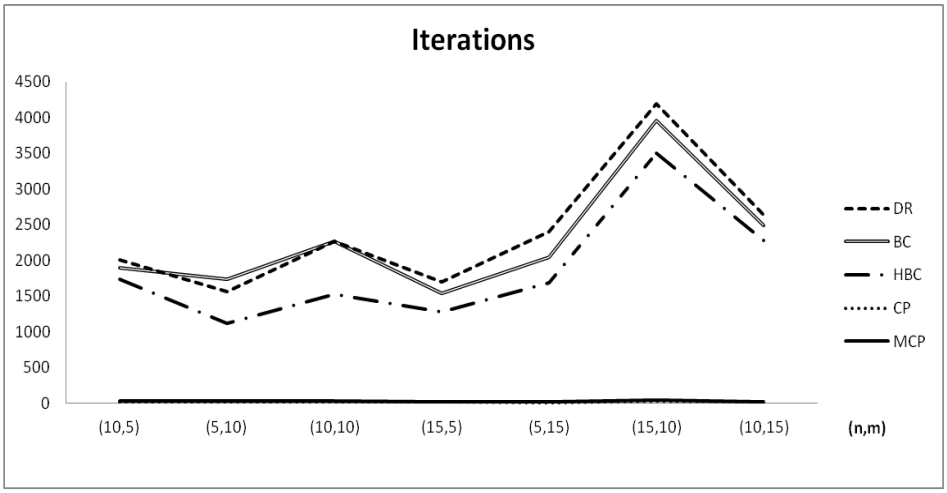
A CPU time reduction of BC and HBC compared to DR is always achieved for all the problems even if it less remarkable than the previous comparison. BC requires less computational time than DR with a difference ranging from -1.57% to -29.44%, while the difference for HBC ranges from -5.71% to -70.63%. Unlike the previous case, computational results clearly show that HBC is the best performing algorithm for each problem

solved. There is no correlation between number of iterations and CPU time: see, for instance, problem with $(n, m) = (10, 10)$ in which DR and BC make almost the same number of iterations, but the CPU time required by BC is significantly smaller. Finally consider the percentage ratio ott/iter: as expected HBC achieves a better ratio because it makes less iterations for solving problem DBLP$''$ especially when the optimal solution is contained in DBLP$'$. In other words HBC is able to halt before BC and this yields a positive impact in terms of efficiency.

A comprehensive comparison among all the algorithms considered is depicted in Figure 4.4. In Tables 4.4 and 4.5 we reported the detail of all the instances solved: note that for HBC there are some instances in which the time limit for solving DBLP$''$ was reached and for the latter the CPU time is omitted and replaced with notation TL. All the algorithms are faster and more efficient than the benchmark and the two cutting plane versions are significantly the best performing ones.



(a) Computational time



(b) Number of iterations

Figure 4.4. Computational comparison

4.1.10 Conclusions

In this section we studied a class of extremely hard problems, Discrete Bilevel Linear Problems (DBLPs). The main features of DBLPs are the bilevel structure, on the one hand, and the discrete nature of the problem, on the other hand. Unlike continuous bilevel linear problems, well known techniques based on polyhedric properties, optimality conditions and reformulations can not be used. Furthermore, the most efficient methods used to solve integer and mixed-integer problems can not be extended in a straightforward manner to DBLPs, since a solution is required not only to be feasible, but also to be bilevel–feasible, that is contained in the follower's reaction set. Starting from these computational difficulties, we developed and presented two new exact algorithms for DBLPs. The first algorithm is based on a cutting plane approach. The rationale behind the algorithm is to solve a relaxation of DBLP in order to compute a lower bound, make a bilevel–feasibility test and then halt if the solution is bilevel–feasible or cut it otherwise. The cut proposed is not a linear function and can be reformulated as a bilevel linear problem in the size of the upper level variables. The resulting problem is a bilevel linear problem with discrete leader's variables and continuous follower's variables. At each iteration, when a cut is added, a new follower problem is included within the original problem, thus the complexity of the problem increases. A modified version of the algorithm tries to remove unnecessary cuts: a positive effect is that the number of follower problems reduces but a negative effect is that a bilevel–feasible solution for a given vector of leader's variables, that has been previously computed, can be found again. Computational tests on a randomly generated set of problems clearly show that both algorithms are faster than a branch and cut existing in the literature used like benchmark. The proposed cutting plane methods are one order of magnitude faster than the benchmark and make a number of iterations two orders of magnitude smaller than the benchmark.

The second algorithm proposed is a branch and cut algorithm that exploits part of the rationale of the benchmark algorithm. The main idea stems from a geometrical property of bilevel problems. Starting from this property we introduce an inequality with the goal of cutting off the largest possible set of integer not bilevel–feasible solutions. We propose a branch and cut algorithm in which the latter inequality is used in the root node as a branching rule for defining two subproblems DBLP$'$ and DBLP$''$ on which the branch and cut is applied separately. While experimenting with our algorithms, we noted that DBLP$'$ is more likely to contain the optimal solution and for this reason it is solved first. Nevertheless, even if the optimal solution is found in DBLP$'$, many iterations of branch and cut on DBLP$''$ have to be performed before certificating the optimality of the solution. In order to reduce the computational time, we propose a hybrid version of the branch and cut in which DBLP$''$ is solved through the cutting plane algorithm above described. A computational comparison on the same test bed previously used indicates a CPU time reduction and a smaller number of iterations for both the algorithms compared to the benchmark. This difference is more notable for the hybrid algorithm. Future research directions may be focused on defining other valid inequalities for DBLPs to incorporate in the proposed cutting plane algorithms which are the most promising exact methods for this class of hard problems.

instance	DR		CP		MCP	
	iter	*CPU*	*iter*	*CPU*	*iter*	*CPU*
10_5_1	2341	239.02	30	44.34	42	46.91
10_5_2	368	35.60	1	1.26	1	1.20
10_5_3	6292	706.49	52	230.68	59	75.54
10_5_4	3117	334.37	28	37.94	44	48.17
10_5_5	1164	126.83	15	19.30	28	31.09
10_5_6	644	64.65	12	15.10	11	13.99
10_5_7	3544	371.13	53	364.73	77	85.32
10_5_8	1708	208.21	48	81.59	71	81.71
10_5_9	244	26.29	3	3.85	4	4.56
10_5_10	646	67.67	8	10.17	9	10.65
5_10_1	738	83.48	15	21.75	18	21.36
5_10_2	2340	529.11	20	26.71	38	41.43
5_10_3	1695	189.09	13	16.91	19	20.69
5_10_4	1491	163.74	29	42.98	49	53.93
5_10_5	4172	994.47	38	58.61	67	75.46
5_10_6	2148	479.19	16	21.37	23	25.05
5_10_7	139	15.87	4	5.09	3	3.70
5_10_8	755	78.03	11	14.04	21	23.60
5_10_9	911	177.68	4	5.54	4	5.54
5_10_10	1286	132.60	18	23.76	28	30.64
10_10_1	1308	300.88	18	24.91	21	25.32
10_10_2	550	104.40	4	5.07	4	5.02
10_10_3	6311	683.75	98	225.73	152	198.64
10_10_4	2908	321.00	32	45.27	38	45.86
10_10_5	1008	199.79	11	14.48	13	16.22
10_10_6	2052	401.93	13	19.22	13	16.74
10_10_7	2812	350.28	18	23.49	19	23.26
10_10_8	1024	220.16	14	18.35	15	19.14
10_10_9	3615	842.87	10	12.64	11	13.18
10_10_10	1050	204.80	16	20.62	25	26.16
15_5_1	938	105.21	8	11.72	8	10.37
15_5_2	1331	135.69	1	1.25	1	1.23
15_5_3	1669	170.31	37	52.90	39	49.22
15_5_4	1439	151.18	20	29.72	22	26.75
15_5_5	2608	267.31	42	64.29	46	54.80
15_5_6	1780	185.98	29	49.80	32	39.62
15_5_7	1872	186.76	8	10.50	9	10.76
15_5_8	801	80.15	10	13.15	9	10.42
15_5_9	1347	161.77	5	6.26	5	6.43
15_5_10	3256	368.49	52	76.19	59	70.81
5_15_1	256	27.67	2	2.28	2	1.95
5_15_2	7365	862.23	16	21.76	24	25.57

instance	DR		CP		MCP	
	iter	*CPU*	*iter*	*CPU*	*iter*	*CPU*
5_15_3	1087	117.19	8	10.55	12	13.07
5_15_4	1443	160.24	4	5.18	5	5.93
5_15_5	300	32.59	3	4.15	3	3.29
5_15_6	3345	360.59	13	18.58	21	23.68
5_15_7	1029	110.34	17	22.26	23	25.94
5_15_8	1247	133.44	7	8.94	9	9.64
5_15_9	7485	823.59	29	46.66	54	68.55
5_15_10	436	45.40	3	3.84	4	4.48
15_10_1	8868	931.27	93	265.28	135	367.90
15_10_2	1177	243.70	7	9.09	9	11.12
15_10_3	45	4.51	1	1.25	1	1.20
15_10_4	5144	549.35	71	137.62	94	112.65
15_10_5	3142	325.31	12	16.91	13	15.57
15_10_6	7548	818.11	43	67.08	75	88.03
15_10_7	4467	469.95	2	2.98	2	2.53
15_10_8	3595	366.68	14	18.49	26	26.41
15_10_9	4997	513.08	44	77.78	67	78.24
15_10_10	2914	298.90	3	4.12	5	5.99
10_15_1	946	97.95	3	3.95	4	5.18
10_15_2	4536	490.64	23	34.38	37	42.46
10_15_3	7172	806.23	48	84.07	71	84.02
10_15_4	1504	195.03	29	48.58	48	54.68
10_15_5	1902	271.52	3	3.96	3	3.81
10_15_6	968	148.17	6	8.16	9	10.41
10_15_7	837	89.22	3	3.99	3	3.85
10_15_8	2361	268.35	15	20.86	18	21.84
10_15_9	5204	573.18	27	42.40	51	58.89
10_15_10	968	208.39	3	3.98	3	3.82

Table 4.4. Computational results in details for DR, CP and MCP

instance	DR			BC			HBC		
	iter	*ott/iter*	*CPU*	*iter*	*ott/iter*	*CPU*	*iter*	*ott/iter*	*CPU*
10_5_1	2341	96.63%	239.02	2094	74.93%	234.30	1670	93.95%	219.74
10_5_2	368	98.91%	35.60	369	86.45%	40.25	341	93.55%	36.60
10_5_3	6292	99.14%	706.49	6293	99.13%	711.50	6293	99.13%	714.57
10_5_4	3117	93.52%	334.37	2938	75.73%	330.36	2474	89.94%	320.38
10_5_5	1164	75.77%	126.83	1193	71.58%	130.73	1121	76.18%	119.64
10_5_6	644	76.09%	64.65	783	68.20%	81.43	669	79.82%	75.58
10_5_7	3544	83.04%	371.13	2533	69.40%	259.99	2332	75.39%	275.48
10_5_8	1708	94.96%	208.21	1179	38.93%	135.24	1121	40.95%	129.89

instance	DR			BC			HBC		
	iter	*ott/iter*	*CPU*	*iter*	*ott/iter*	*CPU*	*iter*	*ott/iter*	*CPU*
10_5_9	244	53.28%	26.29	295	43.39%	32.65	247	51.82%	39.39
10_5_10	646	66.25%	67.67	1360	79.26%	146.27	1117	96.51%	124.60
5_10_1	738	46.48%	83.48	810	78.15%	91.88	782	80.95%	90.53
5_10_2	2340	0.47%	529.11	3382	5.00%	371.89	1324	12.76%	TL
5_10_3	1695	86.08%	189.09	2097	87.98%	244.02	1979	93.23%	225.67
5_10_4	1491	22.87%	163.74	1941	18.65%	218.68	1489	24.31%	182.18
5_10_5	4172	54.60%	994.47	4623	34.59%	584.03	2549	62.73%	TL
5_10_6	2148	98.74%	479.19	1476	29.54%	162.04	967	45.09%	TL
5_10_7	139	63.31%	15.87	234	52.99%	26.72	156	79.49%	19.77
5_10_8	755	95.36%	78.03	802	39.15%	85.44	414	75.85%	55.50
5_10_9	911	93.96%	177.68	528	53.22%	57.74	428	65.65%	58.30
5_10_10	1286	57.23%	132.60	1529	32.96%	163.89	1049	48.05%	131.37
10_10_1	1308	77.52%	300.88	1050	62.76%	125.91	870	75.75%	TL
10_10_2	550	93.82%	104.40	511	86.30%	56.58	394	88.32%	45.75
10_10_3	6311	46.95%	683.75	7028	14.90%	798.28	1696	61.73%	398.13
10_10_4	2908	93.88%	321.00	3338	69.08%	376.74	2742	84.10%	344.95
10_10_5	1008	43.95%	199.79	1279	45.11%	140.73	1161	49.70%	129.01
10_10_6	2052	98.93%	401.93	1465	66.42%	169.88	1185	82.11%	154.21
10_10_7	2812	99.72%	350.28	3060	90.88%	355.09	2927	99.62%	TL
10_10_8	1024	79.79%	220.16	1075	52.65%	120.12	625	90.56%	77.22
10_10_9	3615	98.70%	842.87	2784	90.05%	332.59	2606	96.20%	304.57
10_10_10	1050	2.86%	204.80	1039	87.87%	119.64	1031	88.55%	121.13
15_5_1	938	83.26%	105.21	844	46.92%	98.12	766	51.70%	85.16
15_5_2	1331	99.40%	135.69	600	34.00%	67.64	504	40.48%	54.62
15_5_3	1669	77.71%	170.31	2025	69.58%	228.84	1791	78.67%	196.47
15_5_4	1439	30.09%	151.18	1440	30.07%	155.89	1440	30.07%	154.77
15_5_5	2608	53.37%	267.31	2463	50.95%	269.83	2087	60.13%	245.75
15_5_6	1780	58.48%	185.98	2303	45.72%	251.72	1357	77.60%	175.02
15_5_7	1872	88.94%	186.76	1518	51.12%	161.35	1229	63.14%	TL
15_5_8	801	26.72%	80.15	1271	54.37%	136.19	971	71.16%	114.52
15_5_9	1347	74.46%	161.77	1028	31.91%	106.86	812	40.39%	86.44
15_5_10	3256	98.13%	368.49	1914	79.68%	220.44	1798	84.82%	211.57
5_15_1	256	2.34%	27.67	299	2.34%	33.51	245	2.86%	27.78
5_15_2	7365	95.64%	862.23	3676	69.64%	439.41	3066	83.50%	TL
5_15_3	1087	99.17%	117.19	910	87.91%	99.87	892	89.69%	98.69
5_15_4	1443	97.57%	160.24	2190	98.40%	238.90	1836	100.00%	206.00
5_15_5	300	30.67%	32.59	562	59.25%	59.34	456	73.03%	50.59
5_15_6	3345	79.04%	360.59	3391	30.37%	374.57	2690	38.29%	TL
5_15_7	1029	49.95%	110.34	788	0.89%	85.24	552	1.27%	63.52
5_15_8	1247	72.81%	133.44	541	1.11%	58.70	34	17.65%	12.46
5_15_9	7485	95.07%	823.59	7089	0.18%	779.05	6295	0.21%	TL

instance	DR			BC			HBC		
	iter	*ott/iter*	*CPU*	*iter*	*ott/iter*	*CPU*	*iter*	*ott/iter*	*CPU*
5_15_10	436	88.30%	45.40	1003	94.92%	111.20	775	100.00%	90.53
15_10_1	8868	64.85%	931.27	9395	4.95%	1.079.68	7385	6.30%	881.39
15_10_2	1177	43.08%	243.70	1084	17.71%	126.67	964	19.92%	111.46
15_10_3	45	100.00%	4.51	2	50.00%	1.47	2	50.00%	1.89
15_10_4	5144	78.30%	549.35	3317	84.75%	363.17	2429	54.06%	322.89
15_10_5	3142	97.49%	325.31	3453	68.84%	430.33	2991	79.47%	375.06
15_10_6	7548	91.00%	818.11	6500	65.88%	736.32	5497	77.90%	627.82
15_10_7	4467	98.97%	469.95	4564	97.87%	495.05	4484	99.62%	482.63
15_10_8	3595	97.47%	366.68	3594	97.33%	392.03	3590	97.44%	395.49
15_10_9	4997	97.38%	513.08	4738	70.56%	517.50	4787	84.19%	517.84
15_10_10	2914	67.36%	298.90	2919	67.25%	307.76	2915	67.34%	305.20
10_15_1	946	98.52%	97.95	472	84.96%	47.85	416	96.39%	46.68
10_15_2	4536	84.59%	490.64	5008	85.22%	551.49	4870	87.64%	540.38
10_15_3	7172	88.83%	806.23	5451	96.57%	631.57	4781	82.66%	702.05
10_15_4	1504	31.52%	195.03	1813	18.97%	196.23	1298	23.50%	153.21
10_15_5	1902	92.80%	271.52	2220	94.73%	258.56	2152	97.72%	247.73
10_15_6	968	68.29%	148.17	869	53.62%	98.64	803	58.03%	91.54
10_15_7	837	86.86%	89.22	838	85.80%	90.40	816	88.11%	90.23
10_15_8	2361	96.15%	268.35	2674	91.85%	288.02	2574	95.42%	280.04
10_15_9	5204	64.62%	573.18	3526	60.10%	383.99	3218	65.85%	TL
10_15_10	968	76.76%	208.39	2063	90.40%	227.93	1934	97.00%	211.41

Table 4.5. Computational results in details for DR, BC and HBC

4.2 New heuristic methods for DBLP

In Chapter 3 we introduced a new valid inequality for DCBLP and showed how to compute and use it in order to reformulate the problem on a smaller feasible set and speed up the resolution. The same idea was used in the previous section for developing a branch and cut algorithm for DBLP with positive results. In this section we investigate more in detail the latter inequality for the discrete case, we show that its validity is not always guaranteed for DBLP and, in order to overcome this difficulty, we propose two modified versions of the inequality which are valid under proper necessary and sufficient conditions. Finally, we provide two new fast and efficient heuristics for DBLP based on reformulating the original problem through the addition of the proposed inequalities. Moreover, when the necessary and sufficient conditions hold the heuristics are exact methods and always compute an optimal solution. We also present two special cases in which these conditions are always satisfied and provide experimental results of the proposed heuristics.

4.2.1 Introduction

One of the most relevant results for a generic DBLP has been shown by Moore and Bard [114], who explain why such a problem can not be solved through a pure branch and bound method since some of the classical fathoming

and bounding rules can not be applied. The authors propose a modified branch and bound approach which allows to find an optimal solution if all the variables are discrete and finds a bilevel–feasible solution in the more general mixed-integer case. We have already cited the algorithm proposed by DeNegre and Ralphs [57] based on a simple valid inequality that eliminates an integer solution which is not bilevel–feasible, without modifying the set of all the other integer solutions. The authors use this cut to develop a branch and cut framework which overcomes the computational and theoretical difficulties pointed out in the literature. The algorithm has the advantage to preserve classical fathoming and branching rules, nevertheless the cut used does not take into account the bilevel nature of the problem and removes one by one every integer not bilevel–feasible solution. For this reason the method can be applied only to small size instances due to the computational burden required. Other examples of valid cuts can be found in the literature but only for the linear case (for instance Audet et al. [12] [15] and Shiquan et al. [136]).

4.2.2 Two inequalities to reformulate DBLPs

We have already shown that one of the main criticalities of DBLPs is that relaxing all the integrality requirements the optimal solution computed does not necessarily provide a lower bound. This enables the use of classical branch and bound approaches for discrete problems.

Although the inducible region of a DBLP and the inducible region of its linear relaxation can not be compared from a geometrical standpoint, the solution of this linear relaxation may provide a valid information about the bilevel–feasibility of an integer solution. Recall that it is possible to obtain this kind of information solving the auxiliary problem BLP_{min}^{max} previously defined. Let $(\hat{x}, \hat{y})$ be the optimal solution of BLP_{min}^{max}. We introduce the following two halfspaces

$$\Gamma_y^+ = \{y \mid d^T y \leq \lceil d^T \hat{y} \rceil\}$$

and

$$\Gamma_y^- = \{y \mid d^T y > \lceil d^T \hat{y} \rceil\}$$

For a generic BLP the inequality $d^T y \leq \lceil d^T \hat{y} \rceil$ is a valid cut. Let us call it *bound_inequality*. Note that for every x such that the follower's feasible set $\Omega_y(x)_{BLP}$ is non empty, $R_y(x)$ is one of its extreme points or a facet. It follows that at least an extreme point or a facet of $\Omega_y(x)_{BLP}$ is contained in the halfspace Γ_y^+. More formally

$$\Omega_y(x)_{BLP} \cap \Gamma_y^+ \neq \emptyset \qquad \forall x \text{ such that } \Omega_y(x)_{BLP} \neq \emptyset \tag{4.1}$$

Unfortunately, the extension to DBLP is not immediate and requires some notices. For the sake of clearness, let us call $\Omega_y(x)_{DBLP}$ the follower's feasible region in DBLP and $\Omega_y(x)_{BLP}$ the same region in BLP. As in $\Omega_y(x)_{BLP}$ the follower's variables are not required to be integer, $\Omega_y(x)_{DBLP} \subseteq \Omega_y(x)_{BLP}$. Hence, $\Omega_y(x)_{DBLP}$ can be totally contained in Γ_y^- even if $\Omega_y(x)_{BLP}$ still satisfies condition in equation (4.1): in this case the latter condition does not hold for DBLP. For instance, consider the following problem:

$$
\begin{aligned}
\min_{x,y}\ & F(x,y) = -x - 2y_1 - y_2 \\
\text{s.t.}\quad & x \in \{0,1\} \\
& y \in \min_{y}\ f(y) = y_1 + 2y_2 \\
& \quad \text{s.t.}\quad 6x - 6y_1 - 20y_2 \le -33 \\
& \qquad\quad 6x - 6y_1 + 20y_2 \le 29 \\
& \qquad\quad x + 10y_1 + y_2 \le 35 \\
& \qquad\quad y \in \mathbb{Z}_+
\end{aligned}
$$

The two regions $\Omega_y(x)_{DBLP}$ and $\Omega_y(x)_{BLP}$ are shown in Figure 4.5.

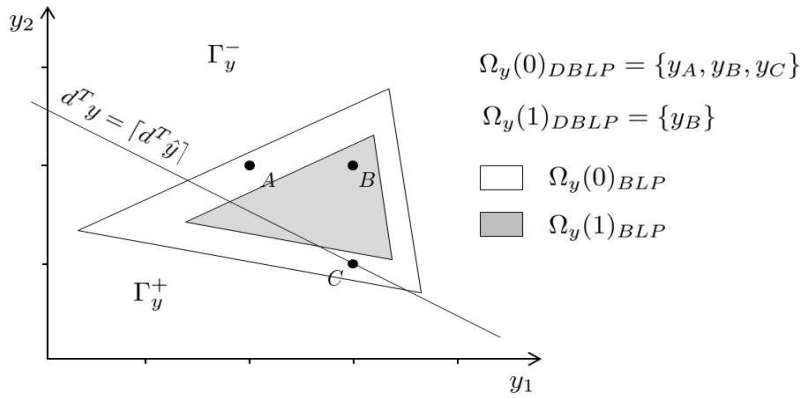


Figure 4.5. Relation between $\Omega_y(x)_{DBLP}$ and $\Omega_y(x)_{BLP}$

The following theorem holds.

Theorem 16. *For a generic DBLP, the inequality $d^T y \le \lceil d^T \hat{y} \rceil$ is a valid cut if and only if, for every x such that $\Omega_y(x)_{DBLP}$ is non empty, there exists at least an integer point of $\Omega_y(x)_{DBLP}$ contained in the halfspace Γ_y^+, i.e.*

$$\Omega_y(x)_{DBLP} \cap \Gamma_y^+ \neq \emptyset \qquad \forall x \text{ such that } \Omega_y(x)_{DBLP} \neq \emptyset \tag{4.2}$$

Proof. (*Necessary*) Let us assume that condition (4.2) is not satisfied. It means that there exists a x such that all integer solutions of $\Omega_y(x)_{DBLP}$ are contained in the halfspace Γ_y^- and hence the rational solution lies in the same halfspace. It follows that there exists an integer bilevel–feasible solution for the DBLP which does not satisfy the inequality $d^T y \le \lceil d^T \hat{y} \rceil$, hence it is not a valid cut.
(*Sufficient*) If condition (4.2) is satisfied, at least one integer solution of $\Omega_y(x)_{DBLP}$ is contained in Γ_y^+ and it dominates all the other possible solutions contained in Γ_y^- which are not bilevel–feasible and can be cut off by the inequality $d^T y \le \lceil d^T \hat{y} \rceil$. Hence it represents a valid cut for DBLPs.
If follows that condition (4.2) is necessary and sufficient to define *bound_inequality* $d^T y \le \lceil d^T \hat{y} \rceil$ a valid cut. □

By Theorem 16 it follows that the validity of *bound_inequality*, is not further assured in the discrete case unless condition (4.2) is verified. In other words, if we reformulate a DBLP through *bound_inequality* we do not have the guarantee that the optimal solution is contained in such reformulation. Thus, integrating *bound_inequality* in an exact method, the resulting approach is a heuristic. Notwithstanding, such integration can be easily

implemented as it works like a preprocessing technique and the goal is to sharp the feasible set and speed up the resolution. Indeed, it usually happens that a DBLP has an elevated number of integer not bilevel–feasible solutions which are very far from the inducible region: the bigger is the distance, the more *bound_inequality* is effective.

Clearly it is very demanding from a computational point of view to verify whether condition (4.2) holds, for this reason it may be convenient to introduce another inequality, obtained by a slight modification of *bound_inequality*, which is less strong but it is more likely to satisfy condition (4.2).

We define the granularity of a given problem as the minimum positive difference between the value of the objective function of two feasible solutions (see Mahajan [104]). Taking into account the follower problem of DBLP, recall that the coefficient vector $d \in \mathbb{Z}^m$. It follows that the granularity of the follower problem is at least the greatest common divisor (GCD) of all the coefficients of d.

Now it is possible to modify *bound_inequality* to introduce another inequality, denoted as *granularity_inequality*, defined as follows

$$d^T y \leq g$$

where g is the smallest multiple of the GCD of all the coefficients of d greater than $d^T \hat{y}$. The computation of *granularity_inequality* is trivial and can be done with the same routine used for *bound_inequality*. Conditions (4.2) are sufficient but not necessary for *granularity_inequality*.

Theorem 17. *If condition (4.2) is satisfied, granularity_inequality is a valid cut for DBLP. The contrary does not necessarily hold.*

Proof. By the definition of g, it is clear that *granularity_inequality* is less strong that *bound_inequality*. In particular $\forall y \in \Gamma_y^+$ we have $d^T y \leq \lceil d^T \hat{y} \rceil \leq g$, hence if $\Omega_y(x)_{DBLP} \cap \Gamma_y^+ \neq \emptyset$, there is at least a rational solution y such that $d^T y \leq g$. Notwithstanding, condition (4.2) is not necessary. In fact there may exist an x such that $\Omega_y(x)_{DBLP} \subset \Gamma_y^-$, hence the rational solution y is such that $d^T y > \lceil d^T \hat{y} \rceil$, but it still satisfies $d^T y \leq g$. Hence the proof. □

This implies that on one side *granularity_inequality* may cut off a smaller portion of the feasible region of a DBLP, but on the other side the halfspace it defines is more likely to contain integer solutions: hence *granularity_inequality* is more likely to be a valid cut compared to *bound_inequality*.

Summing up, we propose two inequalities for DBLPs which are valid under condition (4.2). It is possible to reformulate the original problem adding the proposed inequalities and solve it with a suitable algorithm we introduce in the following section. The method obtained is heuristic for DBLPs. Finally there exist two special cases in which condition (4.2) is always satisfied and the heuristic approach is exact.

4.2.3 Special cases

If the DBLP has only one lower level variable it is easy to show that $d^T y \leq \lceil d^T \hat{y} \rceil$ and $d^T y \leq g$ are valid cuts.

Proposition 10. *If $m = 1$ for a generic DBLP, condition (4.2) is always satisfied.*

Proof. Let us consider an integer vector x. As $m = 1$, $\Omega_y(x)_{BLP}$ is a segment with at least one extreme point in Γ_y^+, denoted with y_1. If $\Omega_y(x)_{BLP}$ has at least one integer point, denoted with y_2, $\Omega_y(x)_{DBLP}$ is not empty.

Let us assume that $y_2 \in \Gamma_y^-$, the convex combination of y_1 and y_2 intersects the equality $d^T y = \lceil d^T \hat{y} \rceil$ in an integer point, denoted with y_3. The integer solution y_3 belongs to $\Omega_y(x)_{BLP}$, because $\Omega_y(x)_{BLP}$ is a convex set for every x, and it belongs to Γ_y^+. Hence, for a given x, there is at least an integer point y_3, which is feasible for the follower, thus belong to $\Omega_y(x)_{DBLP}$, and satisfies the inequality $d^T y \leq \lceil d^T \hat{y} \rceil$. It follows that condition (4.2) holds. □

Note that Proposition 10 is valid only if we round $d^T \hat{y}$ to its ceiling value otherwise if $d^T \hat{y}$ is fractional some bilevel–feasible solutions of the DBLP may be wrongly discarded. Recalling the example of the previous section, in Figure 4.6 a wrong application of the cut is shown which eliminates two only two bilevel–feasible solutions A and B.

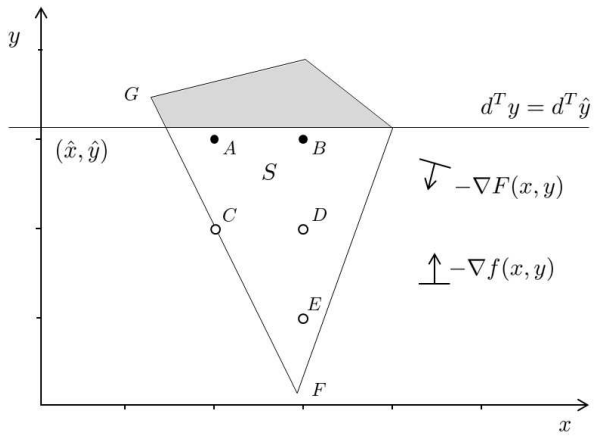


Figure 4.6. Wrong computation of *bound_inequality*

Finally, the following property holds.

Proposition 11. *For a generic DBLP, if matrix B is totally unimodular (TUM) condition (4.2) is always satisfied.*

Proof. The total unimodularity of matrix B implies that the follower's feasible region $\Omega_y(x)_{DBLP}$ is either an empty set or a polyhedron with integer extreme points. It implies that the integrality requirements on follower's variables can be relaxed, the DBLP is equivalent to a DCBLP, hence *bound_inquality* is a valid cut and, by Theorem 16, condition (4.2) holds. □

In other words, if matrix B is TUM, the problem can be reformulated and solved as a DCBLP. Even if in the literature there exist well known techniques to solve DCBLPs, these cuts have never been used to enforce the formulation and reduce the size of the feasible region.

4.2.4 Two new heuristics for DBLP

In this section we describe the two heuristic methods based on the inequalities previously introduced. As pointed out, it is possible to use *bound_inequality* and *granularity_inequality* to reformulate the original problem and then solve it by means of an existing method. In this section we consider the branch and cut method proposed by DeNegre e Ralphs [57] to solve DBLPs as it is well performing and it embeds in a straightforward manner

all the rules of a classical branch and bound approach. For the sake of clearness, this branch and cut method is decsribed in detail since the heuristics we propose are based on a modification of the latter.

A basic fathoming rule for a branch and bound algorithm applied to a discrete problem is that a node k of the branching tree can be pruned if the linear relaxation of the subproblem associated to node k has an integer solution. In a DBLP an integer solution may not be bilevel–feasible which does not allow to apply the same fathoming rule.

In order to overcome this criticality, the rationale behind the algorithm is that every time an integer not bilevel–feasible solution is found it is removed using a valid inequality. The latter is computed as follows.

Assume that the lower level constraints are in form $a_i x + b_i y \leq c_i \quad i = 1 \dots q$.

Proposition 12. *(DeNegre and Ralphs [57]) Let $(\bar{x}, \bar{y})$ be an integer not bilevel–feasible solution of a subproblem k and let I be the subset of lower level constraints active at $(\bar{x}, \bar{y})$. The inequality $\alpha x + \beta y \leq \gamma - 1$, with $\alpha = \sum_{i \in I} a_i$, $\beta = \sum_{i \in I} b_i$ and $\gamma = \sum_{i \in I} c_i$, is satisfied by all the integer solutions of subproblem k but $(\bar{x}, \bar{y})$.*

Finally recall that, as previously explained, given a DBLP it is not sufficient to relax the integrality requirements to compute a valid lower bound. DeNegre and Ralphs propose to relax DBLP removing the integrality constraints, but also dropping the follower's objective function.

Here are the guide lines of the branch and cut method proposed by DeNegre and Ralphs.

Step 0 Initialize $k = 0$ and Z to a sufficiently large number.

Step 1 Solve subproblem k.

- **Step 1.1** (Fathoming) If subproblem k is infeasible prune the current node and go to Step 6.
- **Step 1.2** Let (x_k, y_k) be the optimal solution and go to Step 2.

Step 2 (Fathoming) If $\lceil F(x_k, y_k) \rceil \geq Z$ prune the current node and go to Step 6.

Step 3 (Integrality check) If (x_k, y_k) is integer go to Step 4, else go to Step 5.

Step 4 (Bilevel–feasibility check) Fix x at x_k and solve the follower's problem to compute a bilevel–feasible solution (x_k, y_k^*). If $y_k \neq y_k^*$ go to Step 4.1, else the solution is bilevel–feasible. If $F(x_k, y_k) < Z$ put $Z = F(x_k, y_k)$. Go to Step 6.

- **Step 4.1** (Cut generation) Compute the set I of active constraints at (x_k, y_k) and compute $\alpha = \sum_{i \in I} a_i$, $\beta = \sum_{i \in I} b_i$ and $\gamma = \sum_{i \in I} c_i$. Add the inequality $\alpha x + \beta y \leq \gamma - 1$ to the set of constraints of subproblem k and go to Step 1.

Step 5 (Branching) Select a fractional variable in the solution (x_k, y_k) and branch. Generate 2 subproblems, set $k \leftarrow k + 1$ and go to Step 1.

Step 6 (Backtracking) Select a live node k' if it exists, set $k \leftarrow k'$ and go to Step 1. Otherwise STOP with $F^* = Z$.

The main contribution of such method is that it preserves the basic idea of branch and bound making it applicable to DBLPs and allows to compute an optimal solution only solving linear problems with continuous variables. On the other hand, its main drawback is that the cut generation ignores the geometric properties of DBLPs and it

seems to be quite weak as it removes only a single integer solution in each iteration.

We now describe how it is possible to easily integrate our inequalities in this algorithm in order to obtain two heuristic methods. First of all the GCD of the coefficient vector d is computed for deciding which of the two inequalities to use: if GCD is greater than 1 $granularity\_inequality$ is used in the heuristic, otherwise it is replaced by $bound\_inequality$.

The first heuristic method we present, called *root branch and cut*, integrates the chosen inequality in the original scheme applying it only once in the root node of the branch and bound tree. The goal is to speed up the resolution without dramatically increasing the computational burden. In fact note that the computation of both *bound_inequality* and *granularity_inequality* requires the solution of problem BLP_{min}^{max}, that is a bilevel linear problem. It is clear that if we solved a BLP problem for every node of the branch and bound tree in the worst case we would solve an exponential number of bilevel linear problems. Moreover, the main effects of the new proposed inequalities occur at the very first iterations in which the feasible region is still comprised of lots of integer but not bilevel–feasible solutions. The previous algorithm is modified as follows:

Step 0 Initialize $k = 0$ and Z to a sufficiently large number. Compute GDC. Solve problem BLP_{min}^{max}.

Step 0.1 If BLP_{min}^{max} is infeasible, the original problem is infeasible too, then STOP.

Step 0.2 Let $(\hat{x}, \hat{y})$ be the optimal solution and go to Step 0.3.

Step 0.3 (Inequality generation) If GDC = 1 introduce the inequality $d^T y \leq \lceil d^T \hat{y} \rceil$, otherwise introduce the inequality $d^T y \leq g$, and go to Step 1.

⋮

The second heuristic method, called *extended branch and cut*, selects one of the two inequalities as previously, but in this case it is applied in the root node and every time an integer solution is found which does not satisfy the bilevel–feasibility check. In more detail, we compare the lower level variables of the current solution and that of the corresponding bilevel–feasible solution and apply the new inequality if and only if this difference is greater than a fixed threshold $delta$. The main advantage is that the bilevel–feasibility check has to be made every time an integer solution is found hence it is very simple to compute the “distance" of the current solution from the inducible region. Furthermore, by properly setting the threshold it is possible not to apply this procedure when this distance is small and the inequality may not produce significant effects. As the number of inequality generations may grow exponentially with the number of subproblems, this procedure is applied at most once for each subproblem. In our computational test the threshold δ was set equal to 1. The *extended branch and cut* can be obtained modifying the previous code as follows:

Step 0 Initialize $k = 0$ and Z to a sufficiently large number. Compute GDC. Set threshold $\delta = 1$. Solve problem BLP_{min}^{max}.

Step 0.1 If BLP_{min}^{max} is infeasible, the original problem is infeasible too, then STOP.

Step 0.2 Let $(\hat{x}, \hat{y})$ be the optimal solution and go to Step 0.3.

Step 0.3 (Inequality generation) If GDC = 1 introduce the inequality $d^T y \leq \lceil d^T \hat{y} \rceil$, otherwise introduce the inequality $d^T y \leq g$, and go to Step 1.

⋮

Step 4 (Bilevel–feasibility check) Fix x at x_k and solve the follower's problem to compute a bilevel–feasible solution (x_k, y_k^*). If $y_k \neq y_k^*$ go to Step 4.1, else the solution is bilevel–feasible. If $F(x_k, y_k) < Z$ put $Z = F(x_k, y_k)$. Go to Step 6.

Step 4.1 If $f(y_k) - f(y_k^*) \geq \delta$ and the inequality has not already been added go to Step 4.2, else go to Step 4.3.

Step 4.2 (Inequality generation) If GDC = 1 introduce the inequality $d^T y \leq \lceil d^T \hat{y} \rceil$, otherwise introduce the inequality $d^T y \leq g$, and go to Step 1.

Step 4.3 (Cut generation) Compute the set I of active constraints at (x_k, y_k) and compute $\alpha = \sum_{i \in I} a_i$, $\beta = \sum_{i \in I} b_i$ and $\gamma = \sum_{i \in I} c_i$. Add the inequality $\alpha x + \beta y \leq \gamma - 1$ to the set of constraints of subproblem k and go to Step 1.

⋮

4.2.5 Computational comparison

In this section, we assess the computational performance of the two heuristic methods proposed and compare the results obtained to the exact branch and cut algorithm of DeNegre and Ralphs which is used as benchmark. The rationale behind this comparison is that the three methods make use of the same routine, but the two heuristics solve a reformulated problem. Thus it possible to evaluate the effectiveness of the inequalities proposed in terms of computational elapsed time and quality of the solution.

4.2.6 Numerical results

All the algorithms were applied on a test bed formed by different problems randomly generated following the same rational used in the previous sections. We defined 7 different classes of problem, each one comprised of 10 instances, for a total of 70 instances solved, with $n \in \{5, 10, 15\}$, $m \in \{5, 10, 15\}$ and $q = n + m$. Each parameter was randomly chosen in the range $[-15, 15]$.

The three algorithms were implemented in the C language on a PC Pentium Core 2 Duo with a 2 GHz processor and 1 GB RAM. The AMPL language and the solver CPLEX 12.3 were used to solve mathematical formulations at each tree nodes. The comparison was made taking into account the following features:

- *subprob*, the number of generated subproblems
- *iter*, the number of iterations
- *opt_iter*, the iteration in which the optimal solution was found
- *orig_ineq*, the total number of cuts of DeNegre and Ralphs used by the algorithm
- *new_ineq*, the number of *bound_inequality* or *granularity_inequality* used by the algorithm

- *opt_gap*, the optimality gap between the solution computed by the algorithm and the optimal one
- *CPU*, the required CPU time in seconds spent to solve the DBPL.

We indicate with *native* the original branch and cut method used as benchmark, with *root* our first heuristic and with *extended* our second heuristic.

In Table 4.6 all the results presented are the average values for each class, while in Tables 4.7, 4.8 and 4.9 we reported all the results in detail for all the instances solved.

(n, m)		*subprob*	*iter*	*opt_iter*	*orig_ineq*	*new_ineq*	*opt_gap*	*CPU*
$(10, 5)$	native	1557.00	1717.50	1388.80	160.50	-	-	183.62
	root	1264.40	1397.40	1171.20	133.00	1.00	0.00%	150.80
	extended	1146.60	1261.90	1032.30	46.00	101.10	2.44%	145.14
$(5, 10)$	native	1541.40	1719.40	964.30	178.00	-	-	336.45
	root	1154.20	1238.30	591.50	84.10	1.00	10.14%	228.08
	extended	876.80	977.70	381.70	42.00	105.70	8.71%	182.17
$(10, 10)$	native	1325.20	1434.60	1100.20	109.40	-	-	306.18
	root	1025.80	1101.00	791.10	75.20	1.00	2.50%	232.12
	extended	801.40	851.70	543.30	16.00	34.30	2.50%	179.46
$(15, 5)$	native	1782.20	1809.40	1229.60	81.20	-	-	188.56
	root	1247.20	1303.30	800.90	56.10	1.00	3.08%	140.01
	extended	1048.80	1126.20	718.10	26.80	50.60	4.84%	126.23
$(5, 15)$	native	2704.20	2942.90	2612.10	238.70	-	-	324.77
	root	1880.00	2011.50	998.50	131.50	1.00	6.71%	223.82
	extended	1531.80	1664.30	775.40	34.50	98.00	10.04 %	192.99
$(15, 10)$	native	3371.20	3491.30	2647.30	120.10	-	-	416.25
	root	2797.20	2870.30	1702.80	73.10	1.00	2.00%	351.02
	extended	2716.40	2837.60	1685.10	51.40	69.80	2.00%	349.94
$(10, 15)$	native	3142.20	3312.40	3034.90	170.20	-	-	460.09
	root	2443.80	2537.60	2153.20	93.80	1.00	11.91%	336.35
	extended	2173.90	2295.30	2000.50	43.10	78.20	11.05%	320.97

Table 4.6. Computational results of the three different branch and cut approaches

First of all it is interesting to note that comparing classes $(10, 5)$ and $(5, 10)$, $(15, 5)$ and $(5, 15)$, $(15, 10)$ and $(10, 15)$, in which the number of leader's and follower's variables is inverted, the problems in which $m > n$ always require a higher CPU time for all the three algorithms, except for classes $(15, 10)$ and $(10, 15)$ for *root* and *extended*. This is a reasonable result because the inner complexity of a bilevel problem strictly depends on the size of the lower level problem.

The most relevant result is that *root* and *extended* always outperform *native* analyzing all the key performance indicators we reported. In general, a typical trade-off of branch and cut approaches is that using an increasing number of inequalities the computational burden increases, but the size of the branching tree is reduced. For this reason we propose and compare *root* and *extended* in order to understand the effect of applying multiple inequalities rather than only one inequality in the root node. *Extended* always requires the lowest computational

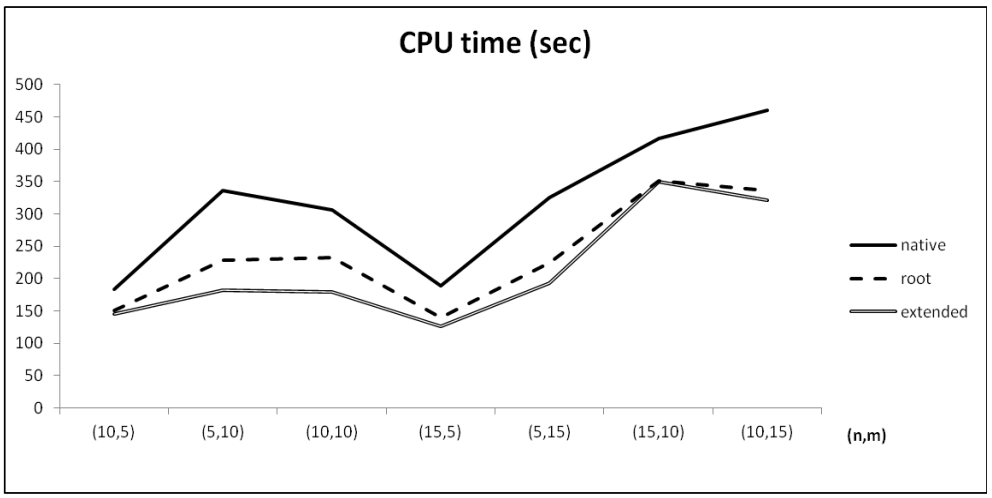


Figure 4.7. Comparison of algorithms in terms of CPU time

time and achieves the smallest number of subproblems and iterations and it is clearly the best performing algorithm in all the classes of test problems. The comparison of the CPU time required by the three algorithms is depicted in Figure 4.7.

Comparing *native* with *extended*, the reduction of generated nodes ranges from 18.72% for class $(15, 10)$ to 43.14% for classes $(5, 10)$ and $(5, 15)$ and the CPU time reduction is still more valuable ranging from 15.93% for class $(15, 10)$ to 45.85% for class $(5, 10)$.

In terms of number of inequalities, note that the application of the new inequalities remarkably reduces the number of the original cuts used. Comparing *native* and *root* the reduction in the number of cuts ranges from 17.13% for class $(10, 5)$ to 52.75% for class $(5, 10)$; it means that the application of the new inequalities in the root node, which can be considered like a preprocessing technique, considerably reduces the number of integer not bilevel–feasible solutions found. This result is even more relevant if we consider *extended*: apart from class $(15, 10)$, it always uses less than half of the number of original cuts compared to *root* with a consequent increase in the number of new inequalities.

Finally, the optimality gap of both the proposed algorithms is never greater than 11.91%. More in detail, *root* solves to optimality 57 out of 70 instances (81.45%) and *extended* gains the same result in 55 test problems (78.57%). The majority of problems in which both the algorithms fail to find the optimal solution are in class $(10, 15)$: in this class the optimality gap of both the algorithms is positive in 4 out of 10 problems with a maximum value equal to 54.17%. It seems that *extended*, which clearly outperform *root* in terms of computational results, is slightly worst in terms of quality of the solution, as expected.

In conclusion, both the heuristics presented are able to improve the effectiveness of the branch and cut approach of DeNegre and Ralphs, simply using two different inequalities to reformulate the original problem. The heuristics find the optimal solution in almost 78.57% of the cases showing that the proposed inequalities have a very positive impact in terms of efficiency and that the negative effects are negligible. These results also suggest the idea that not only the reformulated polyhedron has a reduced size compared to the original one, but it is also very likely to contain the optimal solution.

4.2.7 Conclusions

In the section we propose two new inequalities for a generic DBLP and integrate them within an exact branch and cut algorithm present in the literature. The goal is twofold: on one side we provide two new heuristic methods for DBLPs and on the other side we use the existing branch and cut as benchmark for a performance assessment of our heuristics. By solving an auxiliary bilevel linear problem it is possible to derive two bounds on the value of the follower's objective function and then use this information to reformulate the problem. The proposed inequalities are always valid for BLP, but in the integer case it is only possible to define necessary and sufficient conditions to guarantee their validity. We present two cases in which these conditions are always satisfied.

A computational analysis shows that the use of the two inequalities reduces the number of subproblems, the number of iterations, and also produces a remarkable computational time reduction up to 45.85%. Furthermore, the performance improvement is significantly more valuable than the reduction of solution quality as the computed solution is not the optimum only in 15 out of 70 test problems solved.

The proposed inequalities have to be considered as an initial attempt to improve the resolution of general DBLPs and solve bigger size problems. Future work may be focused on identifying new sufficient conditions for the validity of the proposed inequalities in order to define new cases in which the heuristics are exact solution methods.

instance	*subprob*	*iter*	*opt_iter*	*orig_ineq*	*new_ineq*	*opt_gap*	*CPU*
10_5_1	2041	2341	2262	300	-	-	239.02
10_5_2	361	368	364	7	-	-	35.60
10_5_3	3051	3320	1734	269	-	-	347.77
10_5_4	2805	3117	2915	312	-	-	334.37
10_5_5	235	244	130	9	-	-	24.94
10_5_6	629	644	490	15	-	-	64.65
10_5_7	3271	3544	2943	273	-	-	371.13
10_5_8	1469	1708	1622	239	-	-	208.21
10_5_9	1087	1243	1000	156	-	-	142.82
10_5_10	621	646	428	25	-	-	67.67
5_10_1	709	738	343	29	-	-	83.48
5_10_2	2107	2340	11	233	-	-	529.11
5_10_3	1531	1814	1813	283	-	-	452.62
5_10_4	1357	1491	341	134	-	-	163.74
5_10_5	3581	4172	2278	591	-	-	994.47
5_10_6	1953	2148	2121	195	-	-	479.19
5_10_7	663	724	674	61	-	-	161.24
5_10_8	1437	1570	470	133	-	-	190.38
5_10_9	855	911	856	56	-	-	177.68
5_10_10	1221	1286	736	65	-	-	132.60
10_10_1	1199	1308	1014	109	-	-	300.88
10_10_2	531	550	516	19	-	-	104.40
10_10_3	1665	1814	1806	149	-	-	399.36
10_10_4	1667	1744	766	77	-	-	352.94
10_10_5	985	1008	443	23	-	-	199.79
10_10_6	1967	2052	2030	85	-	-	401.93
10_10_7	179	181	12	2	-	-	34.73
10_10_8	1005	1024	817	19	-	-	220.16
10_10_9	3041	3615	3568	574	-	-	842.87
10_10_10	1013	1050	30	37	-	-	204.80
15_5_1	2869	3075	3074	206	-	-	336.52
15_5_2	1309	1331	1323	22	-	-	135.69
15_5_3	1613	1669	1297	56	-	-	170.31
15_5_4	1733	1812	883	79	-	-	189.46
15_5_5	2487	2608	1392	121	-	-	267.31
15_5_6	1711	1780	1041	69	-	-	185.98
15_5_7	1831	1872	1665	41	-	-	186.76
15_5_8	785	801	214	16	-	-	80.15
15_5_9	2373	2563	967	190	-	-	268.91
15_5_10	571	583	440	12	-	-	64.47
5_15_1	255	256	6	1	-	-	27.67
5_15_2	6755	7555	7410	800	-	-	841.37
5_15_3	2479	2682	2417	203	-	-	301.50

instance	*subprob*	*iter*	*opt_iter*	*orig_ineq*	*new_ineq*	*opt_gap*	*CPU*
5_15_4	1367	1443	1408	76	-	-	160.24
5_15_5	1607	1677	1021	70	-	-	180.15
5_15_6	3109	3345	2644	236	-	-	360.59
5_15_7	953	1029	514	76	-	-	110.34
5_15_8	1167	1247	908	80	-	-	133.44
5_15_9	6941	7485	7116	544	-	-	823.59
5_15_10	2409	2710	2677	301	-	-	308.80
15_10_1	8495	8868	5751	373	-	-	931.27
15_10_2	1169	1177	507	8	-	-	243.70
15_10_3	45	45	45	0	-	-	4.51
15_10_4	4865	5144	4028	279	-	-	549.35
15_10_5	3085	3142	3063	57	-	-	325.31
15_10_6	4413	4560	3046	147	-	-	484.26
15_10_7	1703	1736	1256	33	-	-	179.48
15_10_8	3527	3595	3504	68	-	-	366.68
15_10_9	3201	3278	3071	77	-	-	342.78
15_10_10	3209	3368	2202	159	-	-	735.17
10_15_1	5841	6297	6084	456	-	-	706.09
10_15_2	4377	4536	3837	159	-	-	490.64
10_15_3	3259	3411	3131	152	-	-	366.93
10_15_4	3101	3278	2685	177	-	-	358.04
10_15_5	3509	3707	3523	198	-	-	399.95
10_15_6	2287	2485	2222	198	-	-	288.18
10_15_7	831	837	727	6	-	-	89.22
10_15_8	1313	1352	1164	39	-	-	302.20
10_15_9	5955	6253	6233	298	-	-	1391.29
10_15_10	949	968	743	19	-	-	208.39

Table 4.7. Computational results of the native algorithm

instance	*subprob*	*iter*	*opt_iter*	*orig_ineq*	*new_ineq*	*opt_gap*	*CPU*
10_5_1	1423	1650	1569	227	1	0.00%	167.40
10_5_2	333	340	319	7	1	0.00%	34.09
10_5_3	2745	3025	2487	280	1	0.00%	324.89
10_5_4	2177	2452	2225	275	1	0.00%	277.49
10_5_5	237	244	128	7	1	0.00%	25.96
10_5_6	647	667	534	20	1	0.00%	68.50
10_5_7	2181	2315	1758	134	1	0.00%	238.90
10_5_8	717	798	363	81	1	0.00%	92.18
10_5_9	1283	1517	1398	234	1	0.00%	175.05
10_5_10	901	966	931	65	1	0.00%	103.52

instance	*subprob*	*iter*	*opt_iter*	*orig_ineq*	*new_ineq*	*opt_gap*	*CPU*
5_10_1	863	920	487	57	1	0.00%	100.06
5_10_2	1255	1306	169	51	1	0.00%	295.11
5_10_3	409	465	270	56	1	-50.00%	152.77
5_10_4	1369	1477	362	108	1	0.00%	162.65
5_10_5	3541	3865	3221	324	1	-14.29%	869.92
5_10_6	899	957	436	58	1	0.00%	217.79
5_10_7	575	606	99	31	1	-37.14%	128.93
5_10_8	1241	1325	86	84	1	0.00%	150.85
5_10_9	405	424	281	19	1	0.00%	94.57
5_10_10	985	1038	504	53	1	0.00%	108.17
10_10_1	835	863	655	28	1	0.00%	178.76
10_10_2	469	486	441	17	1	0.00%	97.70
10_10_3	1705	1860	1854	155	1	-25.00%	416.01
10_10_4	999	1009	296	10	1	0.00%	199.54
10_10_5	1117	1160	577	43	1	0.00%	231.15
10_10_6	1131	1172	973	41	1	0.00%	238.62
10_10_7	183	183	12	0	1	0.00%	35.07
10_10_8	607	622	566	15	1	0.00%	137.45
10_10_9	2199	2605	2507	406	1	0.00%	581.12
10_10_10	1013	1050	30	37	1	0.00%	205.81
15_5_1	873	931	769	58	1	-30.77%	106.74
15_5_2	503	503	204	0	1	0.00%	52.43
15_5_3	1713	1790	1409	77	1	0.00%	190.63
15_5_4	1615	1680	787	65	1	0.00%	180.91
15_5_5	1967	2085	1255	118	1	0.00%	223.38
15_5_6	1265	1338	1053	73	1	0.00%	144.25
15_5_7	1213	1224	776	11	1	0.00%	124.75
15_5_8	951	969	691	18	1	0.00%	100.39
15_5_9	2167	2305	965	138	1	0.00%	254.02
15_5_10	205	208	100	3	1	0.00%	22.56
5_15_1	243	244	7	1	1	0.00%	26.47
5_15_2	2629	2883	1661	254	1	-23.33%	315.14
5_15_3	4123	4527	4517	404	1	0.00%	519.22
5_15_4	1205	1233	1188	28	1	-6.25%	140.06
5_15_5	1149	1216	1142	67	1	0.00%	136.13
5_15_6	2513	2683	1030	170	1	0.00%	291.24
5_15_7	531	549	7	18	1	0.00%	59.12
5_15_8	27	28	6	1	1	0.00%	4.12
5_15_9	5945	6287	13	342	1	0.00%	688.96
5_15_10	435	465	414	30	1	-37.50%	57.77
15_10_1	7105	7375	465	270	1	0.00%	806.30
15_10_2	959	963	192	4	1	0.00%	207.56
15_10_3	1	1	1	0	1	0.00%	0.34

instance	*subprob*	*iter*	*opt_iter*	*orig_ineq*	*new_ineq*	*opt_gap*	*CPU*
15_10_4	2343	2385	1313	42	1	0.00%	261.00
15_10_5	2551	2592	2215	41	1	0.00%	278.10
15_10_6	4297	4411	4111	114	1	0.00%	473.04
15_10_7	1703	1736	1264	33	1	0.00%	182.68
15_10_8	3521	3589	3498	68	1	0.00%	379.13
15_10_9	2935	3007	2660	72	1	0.00%	327.15
15_10_10	2557	2644	1309	87	1	-20.00%	594.89
10_15_1	5507	5798	4738	291	1	-20.00%	648.42
10_15_2	4697	4869	4268	172	1	0.00%	525.97
10_15_3	3049	3188	2880	139	1	0.00%	355.76
10_15_4	2083	2161	1474	78	1	0.00%	241.55
10_15_5	2277	2335	2182	58	1	0.00%	255.83
10_15_6	1395	1475	1307	80	1	-54.17%	167.87
10_15_7	809	815	719	6	1	0.00%	89.45
10_15_8	1197	1233	1185	36	1	-26.47%	284.83
10_15_9	1529	1580	914	51	1	-18.42%	374.49
10_15_10	1895	1922	1865	27	1	0.00%	419.34

Table 4.8. Computational results of the root branch and cut

instance	*subprob*	*iter*	*opt_iter*	*orig_ineq*	*new_ineq*	*opt_gap*	*CPU*
10_5_1	683	777	700	3	91	0.00%	80.51
10_5_2	315	322	301	0	7	0.00%	32.09
10_5_3	2745	3297	2729	280	271	0.00%	396.63
10_5_4	2131	2393	2073	8	254	-10.71%	278.98
10_5_5	237	252	135	7	8	0.00%	27.21
10_5_6	647	688	551	20	21	0.00%	73.23
10_5_7	1939	2101	1592	52	110	0.00%	226.68
10_5_8	433	473	191	1	39	0.00%	56.14
10_5_9	1107	996	899	74	134	0.00%	138.81
10_5_10	1229	1320	1152	15	76	-13.64%	141.12
5_10_1	863	977	531	57	57	0.00%	114.25
5_10_2	1255	1353	189	51	47	0.00%	325.96
5_10_3	525	626	536	44	525	-50.00%	149.53
5_10_4	1131	1225	307	15	79	0.00%	141.68
5_10_5	1739	2067	1105	169	159	0.00%	499.42
5_10_6	543	573	275	0	30	0.00%	130.95
5_10_7	575	638	104	31	32	-37.14%	151.23
5_10_8	847	910	55	0	63	0.00%	103.91
5_10_9	305	316	173	0	11	0.00%	71.04
5_10_10	985	1092	542	53	54	0.00%	133.75

instance	*subprob*	*iter*	*opt_iter*	*orig_ineq*	*new_ineq*	*opt_gap*	*CPU*
10_10_1	835	892	679	28	29	0.00%	188.25
10_10_2	469	504	459	17	18	0.00%	107.59
10_10_3	1381	1497	1491	19	97	-25.00%	330.08
10_10_4	999	1021	302	10	12	0.00%	202.18
10_10_5	967	997	446	1	29	0.00%	188.04
10_10_6	1131	1214	1015	41	42	0.00%	255.48
10_10_7	183	184	13	0	1	0.00%	35.51
10_10_8	571	590	534	5	14	0.00%	130.18
10_10_9	465	531	463	3	63	0.00%	126.78
10_10_10	1013	1087	31	36	38	0.00%	230.49
15_5_1	495	607	443	55	57	-30.77%	86.31
15_5_2	503	504	205	0	1	0.00%	51.81
15_5_3	1371	1454	1135	17	66	0.00%	159.78
15_5_4	1615	1744	826	65	64	0.00%	193.86
15_5_5	2153	2394	1696	57	184	-17.65%	276.35
15_5_6	1101	1210	925	53	56	0.00%	137.87
15_5_7	1177	1188	740	0	11	0.00%	123.33
15_5_8	951	987	706	18	18	0.00%	104.80
15_5_9	917	962	402	0	45	0.00%	103.60
15_5_10	205	212	103	3	4	0.00%	24.54
5_15_1	243	246	8	1	2	0.00%	26.50
5_15_2	2005	2205	1101	52	148	-23.33%	254.67
5_15_3	2455	2737	2727	61	221	0.00%	327.35
5_15_4	1197	1228	1183	2	29	-6.25%	141.12
5_15_5	1087	1186	1112	38	61	0.00%	139.85
5_15_6	2513	2847	1097	170	164	0.00%	337.74
5_15_7	1225	1304	71	0	79	-33.33%	145.75
5_15_8	27	29	7	0	2	0.00%	4.59
5_15_9	4131	4376	14	0	245	0.00%	489.23
5_15_10	435	485	434	21	29	-37.50%	63.06
15_10_1	7101	7631	500	270	260	0.00%	866.16
15_10_2	959	968	193	4	5	0.00%	208.93
15_10_3	1	2	2	0	1	0.00%	0.58
15_10_4	2495	2547	1630	1	51	0.00%	281.92
15_10_5	2551	2633	2256	41	41	0.00%	289.77
15_10_6	3865	3968	3684	2	101	0.00%	430.26
15_10_7	1643	1695	1233	18	34	0.00%	183.25
15_10_8	3521	3658	3563	68	69	0.00%	393.45
15_10_9	2935	3079	2730	72	72	0.00%	348.01
15_10_10	2093	2195	1060	38	64	-20.00%	497.08
10_15_1	3951	4195	3998	60	184	-11.43%	484.76
10_15_2	4629	4939	4326	143	167	0.00%	561.69
10_15_3	2486	2604	2300	13	104	0.00%	307.16

instance	*subprob*	*iter*	*opt_iter*	*orig_ineq*	*new_ineq*	*opt_gap*	*CPU*
10_15_4	1699	1770	1129	0	71	0.00%	200.77
10_15_5	2277	2393	2238	58	58	0.00%	271.39
10_15_6	1395	1555	1387	80	80	-54.17%	187.75
10_15_7	809	822	725	6	7	0.00%	94.69
10_15_8	1197	1271	1223	36	38	-26.47%	316.12
10_15_9	1401	1454	788	8	45	-18.42%	339.64
10_15_10	1895	1950	1891	27	28	0.00%	445.69

Table 4.9. Computational results of the extended branch and cut

Chapter 5

New Applications

In this chapter we present two applications of mixed-integer bilevel linear programming to a problem of scheduling and a problem of facility location. Both applications feature the presence of two different decision makers with different objectives, but affecting each other.

5.1 Grid Scheduling by bilevel programming: a heuristic approach

We study the following Grid scheduling problem. A set of independent tasks, submitted to a Grid external scheduler (ES), have to be assigned to a set of Grid computing sites, each one controlled by a local scheduler (LS), for their execution. To each task are associated a release date and a due-date. If the due-date is exceeded, a penalty cost proportional to the tardiness must be paid. If this cost is too high, the ES could prefer to reject the task paying a rejection cost. Indeed, the ES wants to minimize the total cost for rejecting or delaying tasks, while each LS wants to maximize computational resource usage efficiency. Thus problem is modelled by a discrete bilevel programming where the decisions of the ES is constrained by that of the LSs and vice-versa. We propose a heuristic algorithm to solve large size instances and we present and discuss computational results.

5.1.1 Introduction

Grids are distributed computational systems that allow users to access resources of computing sites owned by different organizations (Foster and Kesselman [63]). They have been widely investigated and comprehensive studies within such a framework can be found, e.g., in Casanova and Dongarra [45], Caramia and Giordani [44], Chapin et al. [47], Kapadia and Fortes [82], Litzkow et al. [97], Su et al. [140], Buyya et al. [40],[41].

Grid computing (or the use of a computational Grid) is applying the resources of many computers to user applications that require a great number of computer processing cycles or access to large amounts of data. In this context, Grid scheduling, that is, the allocation of distributed computational resources to user applications, is one of the most challenging and complex task (Nabrzyski et al. [115]).

One of the most known framework for Grid scheduling has been proposed by Ranganathan and Foster [124]. According to this architecture, users submit requests for application (task) execution to the Grid. The latter is modelled by means of three components: an *External Scheduler* (ES) responsible for determining a particular computing site where a submitted task can be executed; a *Local Scheduler* (LS) for each site, responsible for determining the order in which tasks are executed at that particular site; a *Dataset Scheduler*, responsible for determining if and when to replicate data and/or delete local files.

In general, on receipt of a task request, the ES interrogates a set of LSs to ascertain whether the task can be executed on the available local computing resources and meet the user requirements, e.g., the task due-date. If this is the case, a specific computing site that can execute the given task is chosen, and the task request is passed from the ES to this site and is managed locally by the associated LS.

On the basis of this general framework, it is immediate to design a Grid as a two level hierarchical structure. Indeed, in the literature, there exist many examples of Grid scheduling problems in which the decision makers are hierarchically related. In Tchernykh et al. [142] the ES, called Grid broker, allocates tasks to the available computing sites using different scheduling strategies (e.g., minimum load per processor, minimum lower bound on completion time); subsequently the LSs compute the optimal schedule of the assigned tasks. Kurowski et al. [92] [93] propose a similar problem but assume that the ES is not the only decision maker and different stakeholders, with different criteria, have to be taken into account. Also in these works, scheduling at the Grid nodes is made by LSs after the ES assigns the tasks, but the way in which the problem changes according to different local scheduling policies is not investigated.

In this application, we propose a novel modelling framework exploiting the hierarchical structure of Grid scheduling problems. This approach consists in modelling the interactions between the ES and the LSs by means of bilevel optimization. In fact, the ES and the LSs can be modelled as two decision makers whose decisions are hierarchically related, i.e., the decision of the ES (the leader) precedes the decisions of the LSs (the followers) that output their schedules on the basis of the task assignment made by the leader. The resulting model is a DBLP.

As far as we know, this represents the first work in which a bilevel programming formulation is used for such Grid scheduling problems. It implies that the ES and the LSs affect each other though they act sequentially unlike what is modelled in Grid scheduling problems existing in the literature.

We provide a DBLP formulation of the considered Grid scheduling problem and a single level reformulation of the latter. Solving the reformulation by means of a commercial solver (CPLEX) has been shown to be very CPU time demanding and only instances with a few tasks can be optimally solved. Therefore, to cope with real world large size instances, we designed a metaheuristic based on tabu search and exploiting the bilevel model. In an initialization phase the algorithm solves a relaxed version of the single level reformulation and, after fixing the optimal values of the leader's variables so found, it finds the optimal solution of the follower problem, and achieves a bilevel–feasible solution. Then, it keeps on executing iteratively the following three steps: (i) find a new leader's feasible solution by local search holding a tabu list on the moves performed, (ii) solve the follower problem at the optimum fixing the leader's solution found, (iii) compute the bilevel-feasible solution possibly updating the best objective value found so far. The algorithm is halted after a certain number of iterations. To assess the effectiveness of the proposed algorithm, computational results are presented and discussed.

5.1.2 The Grid scheduling framework

In the considered Grid scheduling problem, given a set J of n independent tasks (jobs) submitted to a Grid external scheduler, they have to be assigned by the ES to a set of q Grid computing sites for their execution. For each site s let M_s be the set of processing nodes (machines) that compose site s on which the assigned tasks are locally scheduled by the LS of that site.

For the sake of simplicity, but without loss of generality, we assume that the processing nodes of a site have the same performance (expressed, e.g., in million instructions per unit time period), may access any storage of that site, and can execute at most one task during a given unit time period $[t-1, t)$.

We assume that the (planning) time horizon is given and discretized into τ unit time periods. The total available computational resource units of site s is therefore equal to $|M_s| \cdot \tau$.

Each task j requires a certain amount of computational resources that depends on the task size O_j (expressed, e.g., in million of instructions) and on the performance of the computing site at which it is assigned. Let p_j^s be an integer number representing the computational resource units of site s required by task j if the task is assigned to that site. For example, we can reasonably assume that p_j^s is equal to the ratio between the size of task j and the performance of a processing node of site s and that, without loss of generality, it is an integer value.

It is assumed that tasks are preemptable, more precisely when the execution of a task is interrupted it can be resumed later from its interruption point, but tasks cannot migrate to another site once their execution is started. Moreover, we assume that tasks are malleable, that is, the number of processing nodes assigned to a task may change during its execution.

All tasks are assumed to be known by the Grid system in advance. Task j arrives in the system at a given (release) date r_j and has a due-date d_j, that can be exceeded implying a reduction of the level of service offered by the system to the task owner. The latter induces a penalty cost for the Grid system proportional to the task tardiness, with a penalty cost per unit time period equal to (the weight) w_j of task j. Since this cost reduces the profit of the ES for executing task j, it is reasonable to assume that the tardiness penalty cost is limited by a given maximum amount B_j. If the tardiness penalty cost exceeds B_j it is therefore preferable for the Grid system to reject the task submission with a rejection cost (i.e., the loss of opportunity cost) equal, e.g., to the maximum penalty cost B_j. Therefore, the values of B_j and w_j induce an upper limit $T_j^{\max}$ on the tardiness of task j equal to B_j/w_j, such that if a task could not be executed with at most such tardiness, it would be preferable for the ES to reject the task.

While the ES looks for executing the submitted tasks over the Grid minimizing the total cost for rejecting and delaying tasks, the goal of each LS is maximizing computational resource usage efficiency. In doing this, the LS possibly leaves an over time increasing amount of computational capacity available to answer to other local or external requests of computing. In short, a LS aims as much as possible to have a resource usage profile with high resource usage at the beginning of the time horizon and low usage at the end. We model this LS policy by introducing, for each site s, two resource usage cost parameters u_1^s and u_2^s, and by assuming the cost c_t^{is} of using the resource unit of computational node i of site s in the unit time period $[t-1, t)$ being equal to $(i \cdot u_1^s + t \cdot u_2^s)$. Therefore, the problem of the LS of site s is scheduling all the tasks assigned to that site, minimizing the total cost of the used resources.

The ES problem, together with that of each LS, form a hierarchical optimization problem, where the decision of the ES is constrained by that of the LSs, and vice-versa. In particular, it may be written as a discrete bilevel optimization problem, because the ES (the leader) can only accept (in case paying a task tardiness penalty cost) or reject (with a rejection cost) the tasks, assigning the accepted tasks to the sites, while the LSs (the followers) determine optimal schedules of the assigned tasks without regarding the task tardiness.

5.1.3 A mathematical bilevel formulation

There are a very few papers in the literature where bilevel optimization is used to model (machine) scheduling problems, see, e.g., Karlof and Wang [84], Kis and Kovács [87], Lukaĉ et al. [102]. Note that most of the bilevel models and the solution techniques present in literature use continuous variables despite the use of discrete variables may enable to better model real life problems with an inner combinatorial structure. The difficulties

of bilevel problems in which all the variables or a subset of them are discrete have been well described in the previous chapters.

Next, we give a formulation of the Grid scheduling bilevel problem. To this end, we introduce a dummy site 0, and consider a task as rejected if it is assigned to such a site.

Let us now define the following variables:

- y_j^s is a binary variable, controlled by the ES (i.e., the leader), equal to 1 iff task j is assigned to site s, and 0 otherwise, with $s = 0, 1, \ldots, q$
- T_j is a real non negative variable, controlled by the ES, representing the tardiness of task j
- x_{jt}^{is} is a binary variable, controlled by the LS of site s (i.e., one of the followers), with $s = 1, \ldots, q$, equal to 1 iff task j is assigned (and executed) to a processing node $i \in M_s$ in the (unit) time period $[t-1, t)$, and 0 otherwise.

The leader problem LP is then formulated as follows.

$$(LP) \quad \min_{x,y,T} z_1 = \sum_{j=1}^{n} B_j y_j^0 + \sum_{j=1}^{n} w_j T_j \tag{5.1}$$

$$T_j \geq t \cdot \bar{x}_{jt}^{is} - d_j \qquad \begin{array}{l} j = 1, \ldots, n \\ t = r_j + 1, \ldots, \tau \\ s = 1, \ldots, q \\ i \in M_s \end{array} \tag{5.2}$$

$$\sum_{s=0}^{q} y_j^s = 1 \qquad j = 1, \ldots, n \tag{5.3}$$

$$T_j \geq 0 \qquad j = 1, \ldots, n \tag{5.4}$$

$$y_j^s \in \{0, 1\} \qquad \begin{array}{l} j = 1, \ldots, n \\ s = 0, \ldots, q \end{array} \tag{5.5}$$

where $\bar{x}_{jt}^{is}$ is the optimal solution of the follower problem $FP(s)$ of site s (with $s = 1, \ldots q$). The objective function (5.1) to be minimized is the sum of two contributions: the first one is the total task rejection cost; the latter is the total tardiness cost of the scheduled tasks. Constraints (5.2), together with (5.4), define a lower limit on the tardiness of the tasks as a consequence of the followers' task scheduling decisions. Constraints (5.3), together with (5.5), assure that each task is assigned exactly to one of the q computational sites, or is rejected.

The follower problem $FP(s)$, with $s = 1, \ldots, q$, is formulated in the following, where the dummy task 0 available at time $r_0 = 0$ is introduced for the sake of the formulation.

$$(FP(s)) \quad \min_{x} z_2(s) = \sum_{i \in M_s} \sum_{j=1}^{n} \sum_{t=r_j+1}^{\tau} c_t^{is} x_{jt}^{is} \tag{5.6}$$

$$\sum_{i \in M_s} \sum_{t=r_j+1}^{\tau} x_{jt}^{is} = \begin{cases} p_j^s \, y_j^s & j = 1, \ldots, n \\ |M_s| \cdot \tau - \sum_{k=1}^{n} p_k^s \, y_k^s & j = 0 \end{cases} \tag{5.7}$$

$$\sum_{\{j\in\{0,1,\ldots,n\}|r_j\leq t-1\}} x_{jt}^{is} = 1 \qquad \begin{array}{l} t=1,\ldots,\tau \\ i\in M_s \end{array} \tag{5.8}$$

$$x_{jt}^{is}\in\{0,1\} \qquad \begin{array}{l} j=0,\ldots,n \\ t=r_j+1,\ldots,\tau \\ i\in M_s \end{array} \tag{5.9}$$

The objective function (5.6) to be minimized is the total cost of the used resources of site s. Constraints (5.7) force the number of resource units assigned to task j, with $j=1,\ldots,n$, to be exactly equal to the amount p_j^s of required computational resource units of task j if j is assigned by the leader to that site, and 0 otherwise; moreover, these constraints assign exactly the unused resources of site s to the dummy task 0. Constraints (5.8) assure that, for each unit time period t, computational node i of site s is assigned exactly to one of the tasks (included the dummy) released at time $r_j \leq t-1$.

Note that the overall formulation we propose corresponds to the so called optimistic bilevel model. Recall that, if there exist multiple optimal solutions for problem $FP(s)$ among which the follower is indifferent, it is assumed that the leader can select the solution that optimizes his objective function z_1. This assumption is based on the hypothesis of a semi-cooperative relation among the decision makers.

For the sake of clearness, let us consider a simple instance with the following input data: $n=2$ (tasks 1 and 2), $q=1$ Grid computing site provided with $|M_1|=2$ processing nodes and a planning time horizon of $\tau=4$ time periods. Moreover, task 1 has release-date $r_1=0$, due-date $d_1=1$ and requires $p_1^1=2$ computational resource units of site 1, while for task 2 $r_2=2$, $d_2=4$ and $p_2^1=3$. Let us assume that $y_1^1=1$ and $y_2^1=1$, i.e. tasks 1 and 2 are assigned to site 1 by the leader. Two feasible solutions to problem $FP(s)$ are shown in Figure 5.1.

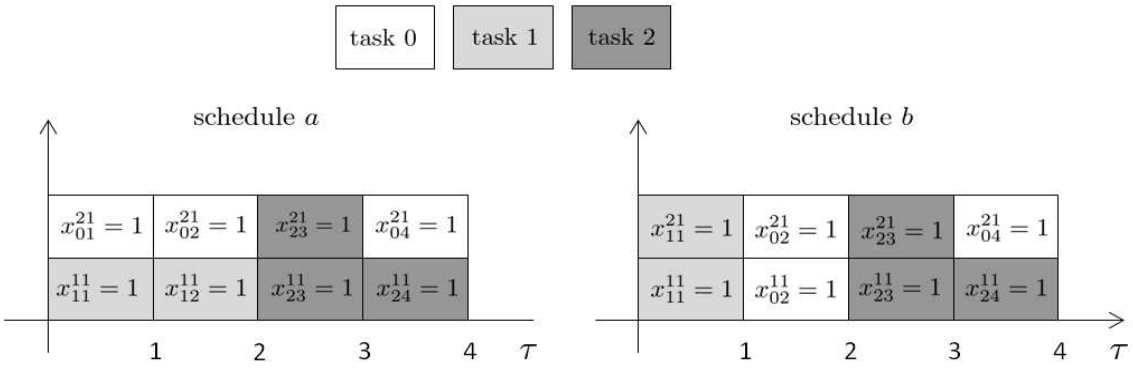


Figure 5.1. Two feasible scheduling solutions for the follower problem

Recall that the cost vector of $FP(s)$ is defined as $c_t^{is}=i\cdot u_1^s+t\cdot u_2^s$, so the follower objective function $z_2(s)$ increases with the time period t and the processing node i. In the example, regardless of the numerical values of the two usage cost parameters u_1^s and u_2^s, the two scheduling solutions in Figure 5.1 are better than any other solutions in the follower's perspective. Comparing schedule a and schedule b, the objective function values are:

$$\begin{array}{ll} \text{schedule } a: & z_2(1)=6u_1^1+13u_2^1 \\ \text{schedule } b: & z_2(1)=7u_1^1+12u_2^1 \end{array}$$

Which is the optimal schedule between a and b depends on the values of u_1^1 and u_2^1: if $u_1^1 > u_2^1$, schedule a is better than schedule b and vice-versa. Note that if the follower optimal solution is schedule a, task 1 has a positive tardiness $T_1 = 1$, while if schedule b is the optimal follower solution both tasks are executed within their due-date. Finally, if $u_1^1 = u_2^1$ the two solutions are equivalent for the follower, but according to the optimistic formulation mentioned above, the leader selects schedule b which does not produce any tardiness.

Problem $FP(s)$ is equivalent to the well known Hitchcock (transportation) problem and, hence, the constraint coefficient matrix is totally unimodular, implying that the optimal solutions of the followers' problems can be obtained by solving their linear relaxations, because the right hand sides of constraints (5.7) are integers. Moreover, $|M_s| \cdot \tau - \sum_{k=1}^{n} p_k^s \, y_k^s + \sum_{j=1}^{n} p_j^s \, y_j^s = \sum_{i \in M_s} \sum_{t=1}^{\tau} 1 = |M_s| \cdot \tau$, i.e., the sum of the right hand sides of constraints (5.7) equals the sum of the right hand sides of constraints (5.8), whatever are the values of the variables $y_k^s \in \{0, 1\}$ chosen by the leader. Therefore, an optimal solution of problem $FP(s)$ always exists given that τ is sufficiently large to guarantee the existence of a feasible solution and also $|M_s| \cdot \tau \geq \sum_{k=1}^{n} p_k^s \, y_k^s$. It follows that the DBLP we presented is equivalent to a DCBLP in which the integrality requirements on the followers's variables are relaxed. This significantly reduces the computational complexity of the model.

5.1.4 The single level reformulation

In this section, we transform the DBLP presented above, that is equivalent to a DCBLP, in its single level reformulation. Since the latter can be optimally solved as linear programs, we can impose the primal feasibility conditions (5.7) and (5.8), the dual feasibility conditions, and the complementary slackness conditions on the followers problems. To this end, let γ_j^s and β_t^{is} be the dual variables associated with constraints (5.7) and (5.8), respectively; since the latter are equality constraints, these dual variables are free in sign. Moreover, let ξ_{jt}^{is} be the non negative slack variables associated to the constraints of the dual of the linear relaxations of problem $FP(s)$, with $s = 1, \ldots, q$. Therefore, we get the following single level reformulation:

$$\min_{x,y,T} z_1 = \sum_{j=1}^{n} B_j y_j^0 + \sum_{j=1}^{n} w_j T_j \tag{5.10}$$

$$T_j \geq t \cdot x_{jt}^{is} - d_j \qquad \begin{array}{l} j = 1, \ldots, n \\ t = r_j + 1, \ldots, \tau \\ s = 1, \ldots, q \\ i \in M_s \end{array} \tag{5.11}$$

$$\sum_{s=0}^{q} y_j^s = 1 \qquad j = 1, \ldots, n \tag{5.12}$$

$$\sum_{i \in M_s} \sum_{t=r_j+1}^{\tau} x_{jt}^{is} = \begin{cases} p_j^s \, y_j^s & j = 1, \ldots, n \\ |M_s| \cdot \tau - \sum_{k=1}^{n} p_k^s \, y_k^s & j = 0 \end{cases} \quad s = 1, \ldots, q \tag{5.13}$$

$$\sum_{\{j \in \{0,1,\ldots,n\} | r_j \leq t-1\}} x_{jt}^{is} = 1 \qquad \begin{array}{l} t = 1, \ldots, \tau \\ s = 1, \ldots, q \\ i \in M_s \end{array} \tag{5.14}$$

$$\gamma_0^s + \beta_t^{is} + \xi_{0t}^{is} = 0 \qquad \begin{array}{l} t = 1, \ldots, \tau \\ s = 1, \ldots, q \\ i \in M_s \end{array} \tag{5.15}$$

$$\gamma_j^s + \beta_t^{is} + \xi_{jt}^{is} = i \cdot u_1^s + t \cdot u_2^s \qquad \begin{array}{l} j = 1, \ldots, n \\ t = r_j + 1, \ldots, \tau \\ s = 1, \ldots, q \\ i \in M_s \end{array} \tag{5.16}$$

$$x_{jt}^{is} \cdot \xi_{jt}^{is} = 0 \qquad \begin{array}{l} j = 0, \ldots, n \\ t = r_j + 1, \ldots, \tau \\ s = 1, \ldots, q \\ i \in M_s \end{array} \tag{5.17}$$

$$x_{jt}^{is} \geq 0 \qquad \begin{array}{l} j = 0, \ldots, n \\ t = r_j + 1, \ldots, \tau \\ s = 1, \ldots, q \\ i \in M_s \end{array} \tag{5.18}$$

$$y_j^s \in \{0, 1\} \qquad \begin{array}{l} j = 1, \ldots, n \\ s = 0, \ldots, q \end{array} \tag{5.19}$$

$$T_j \geq 0 \qquad j = 1, \ldots, n \tag{5.20}$$

$$\xi_{jt}^{is} \geq 0 \qquad \begin{array}{l} j = 0, \ldots, n \\ t = r_j + 1, \ldots, \tau \\ s = 1, \ldots, q \\ i \in M_s \end{array} \tag{5.21}$$

$$\gamma_j^s \quad \text{free} \qquad \begin{array}{l} j = 0, \ldots, n \\ s = 1, \ldots, q \end{array} \tag{5.22}$$

$$\beta_t^{is} \quad \text{free} \qquad \begin{array}{l} t = 1, \ldots, \tau \\ s = 1, \ldots, q \\ i \in M_s \end{array} \tag{5.23}$$

Constraints (5.13)-(5.17) are the KKT optimality conditions for the $FP(s)$ linear relaxations: constraints (5.13) and (5.14) are the primal feasibility conditions, constraints (5.15) and (5.16) are the dual feasibility conditions, and the bilinear constraints (5.17) are the complementary slackness conditions.

In order to linearize constraints (5.17) one may introduce a binary variable δ_{jt}^{is} and replace the bilinear constraints (5.17) as follows

$$\xi_{jt}^{is} \leq M_{jt}^{is} \cdot (1 - \delta_{jt}^{is}) \qquad \begin{array}{l} j = 0, \ldots, n \\ t = r_j + 1, \ldots, \tau \\ s = 1, \ldots, q \\ i \in M_s \end{array} \tag{5.24}$$

$$x_{jt}^{is} \leq \delta_{jt}^{is} \qquad \begin{array}{l} j = 0, \ldots, n \\ t = r_j + 1, \ldots, \tau \\ s = 1, \ldots, q \\ i \in M_s \end{array} \tag{5.25}$$

$$\delta_{jt}^{is} \in \{0,1\} \qquad \begin{array}{l} j = 0,\ldots,n \\ t = r_j+1,\ldots,\tau \\ s = 1,\ldots,q \\ i \in M_s \end{array} \tag{5.26}$$

where M_{jt}^{is} are big-M constants. W.l.o.g., we can assume M_{jt}^{is} to be equal to an upper bound on the optimal value of the slack variable ξ_{jt}^{is}, and hence $M_{jt}^{is} \leq (i \cdot u_1^s + t \cdot u_2^s) + 2U^s$, for $j = 1,\ldots,n$, and $M_{0t}^{is} \leq 2U^s$, with $U^s = [(n+1) + (|M_s| \cdot \tau) - 1](|M_s| \cdot u_1^s + \tau \cdot u_2^s)$ being an upper bound on the absolute value of the optimal dual variables γ_j^s and β_t^{is} of the follower problems $FP(s)$, with $s = 1,\ldots,q$, which, we recall, are equivalent to the transportation problem [1].

Hence, the overall resulting mixed-integer formulation of the single level reformulation, denoted as SLF, is the following:

$$(SLF) \quad \min_{x,y,T} z_1 = \sum_{j=1}^{n} B_j y_j^0 + \sum_{j=1}^{n} w_j T_j \tag{5.27}$$

$$T_j \geq t \cdot x_{jt}^{is} - d_j \qquad \begin{array}{l} j = 1,\ldots,n \\ t = r_j+1,\ldots,\tau \\ s = 1,\ldots,q \\ i \in M_s \end{array} \tag{5.28}$$

$$\sum_{s=0}^{q} y_j^s = 1 \qquad j = 1,\ldots,n \tag{5.29}$$

$$\sum_{i \in M_s} \sum_{t=r_j+1}^{\tau} x_{jt}^{is} = \begin{cases} p_j^s \, y_j^s & j = 1,\ldots,n \\ |M_s| \cdot \tau - \sum_{k=1}^{n} p_k^s \, y_k^s & j = 0 \end{cases} \quad s = 1,\ldots,q \tag{5.30}$$

$$\sum_{\{j \in \{0,1,\ldots,n\} | r_j \leq t-1\}} x_{jt}^{is} = 1 \qquad \begin{array}{l} t = 1,\ldots,\tau \\ s = 1,\ldots,q \\ i \in M_s \end{array} \tag{5.31}$$

$$\gamma_0^s + \beta_t^{is} + \xi_{0t}^{is} = 0 \qquad \begin{array}{l} t = 1,\ldots,\tau \\ s = 1,\ldots,q \\ i \in M_s \end{array} \tag{5.32}$$

$$\gamma_j^s + \beta_t^{is} + \xi_{jt}^{is} = i \cdot u_1^s + t \cdot u_2^s \qquad \begin{array}{l} j = 1,\ldots,n \\ t = r_j+1,\ldots,\tau \\ s = 1,\ldots,q \\ i \in M_s \end{array} \tag{5.33}$$

[1] Any basic feasible solution of the transportation problem with N origins and M destinations and costs ρ_{hk} (with $h = 1,\ldots,N$ and $k = 1,\ldots,M$) has exactly $(N+M-1)$ basic variables whose values can be non-zero, while the other (non-basic) variables are equal to zero. Denoting with u_h, v_k the dual variables, by applying the complementary slackness conditions $u_h + v_k = \rho_{hk}$ for each non-zero basic variable of the basic feasible solution, and assuming $u_1 = 0$, we can determine the values of the dual variables u_h, v_k related to the primal basic feasible solution. Since the number of such conditions are at most as many as the number of basic variables (i.e., $N+M-1$), we have that $|u_h| \leq (N+M-1)\max\{\rho_{hk}\}$ (an analog consideration follows for the bound on the values of variables v_k)

$$\xi_{jt}^{is} \leq M_{jt}^{is} \cdot (1 - \delta_{jt}^{is}) \quad \begin{array}{l} j = 0, \ldots, n \\ t = r_j + 1, \ldots, \tau \\ s = 1, \ldots, q \\ i \in M_s \end{array} \tag{5.34}$$

$$x_{jt}^{is} \leq \delta_{jt}^{is} \quad \begin{array}{l} j = 0, \ldots, n \\ t = r_j + 1, \ldots, \tau \\ s = 1, \ldots, q \\ i \in M_s \end{array} \tag{5.35}$$

$$x_{jt}^{is} \geq 0 \quad \begin{array}{l} j = 0, \ldots, n \\ t = r_j + 1, \ldots, \tau \\ s = 1, \ldots, q \\ i \in M_s \end{array} \tag{5.36}$$

$$y_j^s \in \{0, 1\} \quad \begin{array}{l} j = 1, \ldots, n \\ s = 0, \ldots, q \end{array} \tag{5.37}$$

$$T_j \geq 0 \quad j = 1, \ldots, n \tag{5.38}$$

$$\xi_{jt}^{is} \geq 0 \quad \begin{array}{l} j = 0, \ldots, n \\ t = r_j + 1, \ldots, \tau \\ s = 1, \ldots, q \\ i \in M_s \end{array} \tag{5.39}$$

$$\delta_{jt}^{is} \in \{0, 1\} \quad \begin{array}{l} j = 0, \ldots, n \\ t = r_j + 1, \ldots, \tau \\ s = 1, \ldots, q \\ i \in M_s \end{array} \tag{5.40}$$

$$\gamma_j^s \quad \text{free} \quad \begin{array}{l} j = 0, \ldots, n \\ s = 1, \ldots, q \end{array} \tag{5.41}$$

$$\beta_t^{is} \quad \text{free} \quad \begin{array}{l} t = 1, \ldots, \tau \\ s = 1, \ldots, q \\ i \in M_s \end{array} \tag{5.42}$$

Theoretical properties

In this subsection we study some theoretical properties on the SLF. Let us denote with $R_x(y_j^s)$ the set of optimal solutions of the linear relaxation of $FP(s)$.

Recalling that: (i) the constraint coefficient matrix of the follower problem is totally unimodular and the right hand sides are integer values; (ii) there are upper level constraints which do not only depend on the upper level variables, we can show the following results.

Proposition 13. *If one of followers problems, say $FP(s)$, has multiple optimal solutions for a given vector y_j^s, $R_x(y_j^s)$ is formed by at least two integer vertices of the polyhedron of $FP(s)$ and by their convex combination.*

Proof. $FP(s)$ is an integer linear programming problem that, as we previously noted, may be solved by its linear relaxation due to the total unimodularity of the constraint coefficient matrix and the integrality of the right

hand sides. It follows that, for a given vector y_j^s, if $FP(s)$ has a unique optimal solution, the latter must be integer and this is also the unique solution of the linear relaxation of $FP(s)$; otherwise, $R_x(y_j^s)$ is a face of the polyhedron of $FP(s)$, defined by at least two (integer) optimal vertices and their convex combination. □

Proposition 14. *If the optimal solution (y_j^{s*}, x_{jt}^{is*}) of problem SLF has fractional lower level variables, at least one of the followers problems, say $FP(s)$, has multiple optimal solutions given vector y_j^{s*}, and vector x_{jt}^{is*} belongs to a face of the polyhedron of $FP(s)$.*

Proof. If variables x_{jt}^{is*} are fractional, it happens that: (i) problem $FP(s)$ has multiple optimal solutions and, by Proposition 13, $R_x(y_j^{s*})$ is a face of the polyhedron of $FP(s)$; (ii) upper level constraints (5.28) and (5.38) in SLF intersect $R_x(y_j^{s*})$ defining a new vertex, lying on a face of the feasible region of $FP(s)$. The latter vertex is the optimal fractional solution of problem SLF. □

Note that the only constraints which depend on the follower's variables are constraints (5.28) which define the tardiness of the scheduling problem together with constraints (5.38) which impose the non negativity of the tardiness. It is important to underline that constraints (5.28) and (5.38) do not impose any feasibility condition on variables x_{jt}^{is} but they are only necessary to linearize the leader's objective function. It follows that all the integer solutions contained in the set $R_x(y_j^{s*})$ are bilevel–feasible.

Since the tardiness is defined as a non negative variable, it happens that among all the integer solutions contained in $R_x(y_j^{s*})$, some correspond to a solution with $T_j \geq \alpha_j$, with $j = 1, \ldots, n$ and $\alpha_j \geq 0$, and at least one vertex corresponds to a solution with $T_j \geq \alpha'_j$, with $\alpha'_j < 0$ for some $j = 1, \ldots, n$. Therefore, by the intersection of $R_x(y_j^{s*})$ with constraints (5.28) and (5.38), a new fractional basic feasible solution $(y_j^{s*}, \bar{x}_{jt}^{is})$ of SLF is defined where $\bar{x}_{jt}^{is}$ is a convex combination of the integer solutions in $R_x(y_j^{s*})$ and yields the smallest value for the leader's objective function.

Another simplest way to understand Proposition 14 is that the leader's objective function can be rewritten as a non linear function removing constraints (5.28) as follows:

$$\min_{x,y,T} z_1 = \sum_{j=1}^{n} B_j y_j^0 + \sum_{j=1}^{n} w_j \cdot \max_{t,i,s}[t \cdot x_{jt}^{is} - d_j, 0]$$

In this case the optimal solution may be an internal solution of $R_x(y_j^{s*})$ which is a face of the polyhedron of $FP(s)$.

The latter result implies that if we solve SLF we are not guaranteed that the lower level variables will be integer.

Proposition 15. *If the optimal solution (y_j^{s*}, x_{jt}^{is*}) of problem SLF has fractional lower level variables for at least one site, say site s, there is always a solution with integer lower level variables which is bilevel–feasible and can be easily computed.*

Proof. By Proposition 14 we know that if x_{jt}^{is*} is fractional, it is a convex combination of the integer solutions in $R_x(y_j^{s*})$ which are all bilevel–feasible; at least one of them can be computed by fixing the upper level variables y_j^{s*} and solving the follower problem $FP(s)$. □

Consider a simple example in which there are only $n = 2$ tasks, $q = 1$ site, $|M_s| = 1$ processing node and a time horizon $\tau = 3$. Moreover, consider the following data: $B_1 = B_2 = 10$ (they are large enough to avoid task rejection), $u_1 = u_2 = 1$, $d_1 = d_2 = 2$, $p_1 = 1$, $p_2 = 2$, $w_1 = 2$, $w_2 = 1$, $r_1 = r_2 = 0$. Clearly it is not convenient for the leader to reject the tasks and so task 1 and task 2 are scheduled on the same site and on the same processing node. This implies that $y_j^{1*} = 1$ and $y_j^{0*} = 0$ for $j = 1, 2$.

There are three optimal integer solutions for $FP(s)$, let us call them A, B and C as represented in Figure 5.2.

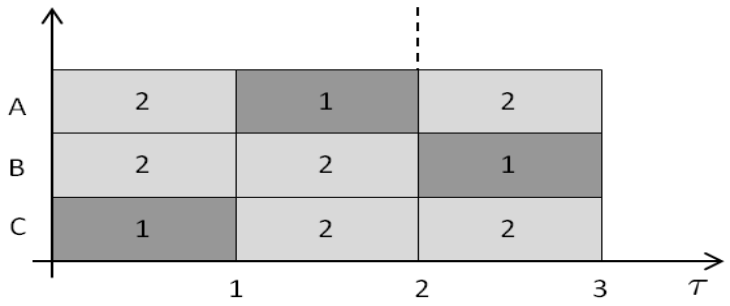


Figure 5.2. Optimal integer solutions for the follower problem

For solution A we have the following upper level constraints (5.28): $T_1 \geq 0$, $T_2 \geq 1$ and a value for the leader's objective function equal to 1, while for solution B we have: $T_1 \geq 1$, $T_2 \geq 0$ and a value for the leader's objective function equal to 2. Let us now consider solution C. The upper level constraints (5.28) are: $T_1 \geq -1$, $T_2 \geq 1$. Despite the previous two solutions, for solution C the non negativity constraint (5.38) on the tardiness $T_1 \geq 0$ is not redundant and intersect the set $R_x(y_j^{s*})$ defining a new fractional solution D which is a convex combination of A, B and C. The solution to the leader problem corresponding to the follower's solution C has $T_1 = 0$, $T_2 = 1$ with an objective function value equal to 1.

By solving problem SLF its optimal solution corresponds to the following fractional solution D (apexes $i = 1$ and $s = 1$ are omitted for the sake of clearness):

$$\begin{pmatrix} x_{01} = 0 & x_{11} = \frac{1}{3} & x_{21} = \frac{2}{3} \\ x_{02} = 0 & x_{12} = 0 & x_{22} = 1 \\ x_{03} = 0 & x_{13} = \frac{2}{3} & x_{23} = \frac{1}{3} \end{pmatrix}$$

for which $T_1 = T_2 = 0$ and the optimal value for the leader's objective function is 0. It follows, by the above propositions, that if we solve SLF we may obtain a non integer solution, but for the same values of the upper level variables there will always be a solution which is integer and optimal for the follower. Hence, it is possible to solve SLF, then fix the upper level variables, and finally solve $FP(s)$ obtaining an integer bilevel–feasible solution. Based on these observations, we developed a heuristic algorithm to solve the linearized single level reformulation SLF which is described in the following.

5.1.5 The heuristic algorithm

It is well known in bilevel optimization that the single level reformulation obtained by a bilevel program is very difficult to solve because of the introduction of many additional binary variables. Another criticality of the SLF, as previously explained, is that there is no guarantee of finding an optimal integer solution. The simplest way to overcome this problem, is to modify the formulation of problem SLF resetting variables x_{jt}^{is} to be binary.

In order to see the limits of SLF problem in terms of size of solvable instances, we started by conducting preliminary tests of problem SLF, by implementing and solving the latter in the AMPL language and with

CPLEX 12.3, on a PC Pentium Core 2 Duo with a 2 GHz processor and 1 GB RAM, respectively. The results of these tests were, as expected, not satisfactory for the application under consideration. In particular, considering the test case with $q = 5$ identical computing sites, each one with $|M_s| = 3$ machines, CPLEX was able to solve problem SLF at the optimum within a reasonable computing time for the application (we imposed a time limit of two minutes) only on instances with at most five tasks. The outcome of this experimentation led us to look for a heuristic algorithm able to treat larger instances in limited computing time. This algorithm is based on tabu search and is described in the following.

Initialization phase: solution of problem $RSLF$ and integrality test. In the first phase we solve a relaxation of problem SLF that is made by means of three steps: (i) we replace the equality constraints (5.30) and (5.31) of the follower problem with inequality constraints; (ii) we remove the constraints (5.32)-(5.35) and (5.40) concerning with the optimality of the follower problem; (iii) we add constraints (5.48) in order to avoid scheduling task j on site s if j is not assigned to s. Hence, we solved the following relaxed single level reformulation $RSLF$

$$(RSLF) \quad \min_{x,y,T} z_1 = \sum_{j=1}^{n} B_j y_j^0 + \sum_{j=1}^{n} w_j T_j \tag{5.43}$$

$$T_j \geq t \cdot x_{jt}^{is} - d_j \qquad \begin{array}{l} j = 1,\ldots,n \\ t = r_j + 1,\ldots,\tau \\ s = 1,\ldots,q \\ i \in M_s \end{array} \tag{5.44}$$

$$\sum_{s=0}^{q} y_j^s = 1 \qquad j = 1,\ldots,n \tag{5.45}$$

$$\sum_{i \in M_s} \sum_{t=r_j+1}^{\tau} x_{jt}^{is} \geq \begin{cases} p_j^s \, y_j^s & j = 1,\ldots,n \\ |M_s| \cdot \tau - \sum_{k=1}^{n} p_k^s \, y_k^s & j = 0 \end{cases} \quad s = 1,\ldots,q \tag{5.46}$$

$$\sum_{\{j \in \{0,1,\ldots,n\} | r_j \leq t-1\}} x_{jt}^{is} \geq 1 \qquad \begin{array}{l} t = 1,\ldots,\tau \\ s = 1,\ldots,q \\ i \in M_s \end{array} \tag{5.47}$$

$$\sum_{i \in M_s} \sum_{t=r_j+1}^{\tau} x_{jt}^{is} \leq y_j^s \cdot |M_s| \cdot \tau \qquad \begin{array}{l} s = 1,\ldots,q \\ j = 1,\ldots,n \end{array} \tag{5.48}$$

$$x_{jt}^{is} \geq 0 \qquad \begin{array}{l} j = 0,\ldots,n \\ t = r_j + 1,\ldots,\tau \\ s = 1,\ldots,q \\ i \in M_s \end{array} \tag{5.49}$$

$$y_j^s \in \{0,1\} \qquad \begin{array}{l} j = 1,\ldots,n \\ s = 0,\ldots,q \end{array} \tag{5.50}$$

$$T_j \geq 0 \qquad j = 1,\ldots,n \tag{5.51}$$

We compute a solution ($\bar{y}_j^s$,$\bar{x}_{jt}^{is}$) and make an integrality test on variables $\bar{x}_{jt}^{is}$: if $\bar{x}_{jt}^{is}$ are binary and satisfy the optimality conditions for the followers' problems, solution ($\bar{y}_j^s$,$\bar{x}_{jt}^{is}$) is optimal for SLF and hence optimal for

the optimistic version of the bilevel problem. Otherwise, we start our heuristic approach based on a tabu search mechanism.

Iterative phase: tabu search. In this phase a tabu search mechanism is developed. The idea is similar to that applied by Wen and Huang [150] for a generic DCBLP. Differently from [150], in the proposed algorithm we do not use any diversification technique and even the mechanism of local search is quite different because we use specific information connected to the model meaning. Moreover, our heuristic is thought to be applied to large instances unlike that of Wen and Huang.

The main objective of a tabu search is to find a local optimum and its basic scheme is the following: compute an initial solution, compute a good neighbor solution by a local search in order to try to improve the objective function value and then move from it. Moving from a solution to another one is allowed even if there is no improvement in the objective function: in this case the move is classified as tabu and is recorded in a tabu list to avoid returning to a previously computed better solution and try to explore solutions far from the current local optimum.

The first iteration starts from solution $(\bar{y}_j^s,\bar{x}_{jt}^{is})$ calculated in the initialization phase. The algorithm fixes the upper level variables to $\bar{y}_j^s$ and computes an integer rational solution for the followers by solving $FP(s)$. Let $(\bar{y}_j^s,\hat{x}_{jt}^{is})$ be such a solution. From this initial solution the local search mechanism, described in the following, starts. The algorithm computes the weighted tardiness values $w_j\bar{T}_j$ and $w_j\hat{T}_j$ corresponding to solutions $(\bar{y}_j^s,\bar{x}_{jt}^{is})$ and $(\bar{y}_j^s,\hat{x}_{jt}^{is})$, respectively. Note that tasks weighted tardiness are part of the leader's objective function and they are crucial to determine the quality of the solution. Let

$$j^* \in argmax\{(w_j\hat{T}_j - w_j\overline{T}_j)\}$$

where index j^* corresponds to the task for which there is the highest difference of the weighted tardiness between the integer and the fractional solution. Let s^* the site in which task j^* is assigned, i.e. $\bar{y}_{j^*}^{s^*} = 1$. Once j^* is computed, the algorithm forbids the assignment of task j^* to site s^* (local move). Hence we solve $RSLF$ fixing variables $y_{j^*}^{s^*} = 1 - \bar{y}_{j^*}^{s^*}$, so obtaining a solution $(\bar{y}_j^s,\bar{x}_{jt}^{is})$ from which a new bilevel-feasible solution $(\bar{y}_j^s,\hat{x}_{jt}^{is})$ is computed by solving problems $FP(s)$. If the new feasible solution is worse than the best found so far, the local move used is recorded as tabu in a proper tabu list; the number max_tabu representing the number of iterations within which a tabu move is forbidden is a parameter of the algorithm (this number is decreased by one at each iteration until the move returns eligible). The algorithm keeps on iterating until one of the following stopping criteria is met:

1. a predefined number max_iter of consecutive non improving iterations of the tabu search have been executed;
2. a time limit is reached.

5.1.6 Simulated scenarios

In this section, we describe the experimentation conducted on the proposed heuristic. Our implementation environment included the following: a PC Pentium Core 2 Duo with a 2GHz processor and 1GB RAM, a Microsoft Windows C++ Developer 6.0 compiler, the AMPL language, and the CPLEX 12.3 solver.

In recent work concerning with Grid scheduling problem the performance and the general behaviour of models and proposed algorithms are tested on real data. Iosup et al. [77] realized the project of a Grid Workload Archive (GWA) in which real workloads coming from different Grid environments are collected and stored in a predefined format. According to Kurowski et al. [92] [93], even if a computational analysis on real data is more significative in theory, the use of real workloads implies some criticality. Only a few instances provide all the necessary information required to evaluate a model and its solving algorithm (e.g., due-dates or penalty costs) and a real workload coming from a Grid system may not be used as a valid benchmark on another system due to the presence of specific constraints and policies.

For this reason we conducted two separate simulations, both on synthetic and real workloads. We performed both the simulations on a fictitious Grid system formed by $q = 5$ sites with the same characteristics. In particular, each site s is assumed to be composed of $|M_s| = 3$ processing nodes (machines), with the same speed equal to 500 MI per time unit. The parameters u_1^s and u_1^s appearing in the cost function $c_t^{is} = i \cdot u_1^s + t \cdot u_2^s$ are assumed to be equal to 1.

Synthetic workloads

The synthetic workloads were created in the following manner. Tasks arrive according to a Poisson arrival process, with expected arrival rate (i.e., number of tasks per time unit) equal to λ.

Task sizes O_j (in MI) are uniformly generated at random with an expected value $E[O_j] = 10000$ MI, which is similar to the order of magnitude of the task size assumed in Buyya [40] and with minimum value equal to $0.8 \cdot E[O_j]$ and maximum value equal to $1.2 \cdot E[O_j]$. This implies that the computational resource units p_j^s required by j on site s is on average equal to 20. Moreover, since the sites are assumed to be the same, the computational resource units p_j^s required by j is the same for each site s.

Task due-dates d_j are uniformly generated at random with an expected value $E[d_j] = r_j + run\_time$, a minimum value equal to $r_j + 0.8 \cdot (run\_time)$, and a maximum value equal to $r_j + 1.2 \cdot (run\_time)$, where r_j is the task arrival date. The parameter run_time (posed equal to 10 time units) is approximately the expected task run time, assuming that, on average, $\frac{2}{3}$ of the computational resources of a site is allocated to the task during the run time period.

The maximum tardiness penalty costs B_j, i.e., the task rejection costs, are uniformly generated at random with a given expected value $E[B_j]$, a minimum value equal to $0.8 \cdot E[B_j]$, and a maximum value equal to $1.2 \cdot E[B_j]$.

Finally, the tardiness penalty unitary costs (task weights) w_j, are uniformly generated at random with a given expected value $E[w_j] = 10$, a minimum value equal to $0.8 \cdot E[w_j]$ and a maximum value equal to $1.2 \cdot E[w_j]$.

All the generated data values are non negative integers. We have experimented with $\lambda = 1, 2, 3, 4, 5$, and $E[B_j] = 50, 100, 150$, considering $n = 100$ tasks. The number of runs of the tabu search routine embedded in the heuristic algorithm was fixed to $max\_iter = 15$, the lenght of the tabu list was fixed to $max\_tabu = 5$ and the time limit is 300 second. Moreover, we imposed a time limit of two minutes for the solution of every $RSLF$ and $FP(s)$ problems. The number τ of unit time periods of the (planning) time horizon was assumed sufficiently large.

The results are listed in Tables 5.1-5.3. In the tables, the columns list the average task arrival rate λ, the average rejection cost, the average total cost among the 100 tasks, the average tardiness cost of the (accepted) scheduled tasks, the number of rejected tasks and the CPU time in seconds spent by the algorithm.

λ	avg_rej_cost	avg_tot_cost	avg_tard_cost	rej_tasks	CPU
1	6.75	11.15	5.18	15	75.48
2	39.48	40.04	4.31	87	166.59
3	34.61	42.04	33.77	78	124.75
4	0.00	21.92	21.92	0	96.72
5	45.71	45.71	0.00	100	108.71

Table 5.1. Synthetic workloads with $E[B_j] = 50$

λ	avg_rej_cost	avg_tot_cost	avg_tard_cost	rej_tasks	CPU
1	0.00	0.11	0.11	0	55.39
2	53.94	59.83	14.37	59	124.33
3	94.97	94.97	0.00	99	248.65
4	42.58	87.44	84.64	47	124.35
5	55.08	69.61	35.44	59	124.60

Table 5.2. Synthetic workloads with $E[B_j] = 100$

λ	avg_rej_cost	avg_tot_cost	avg_tard_cost	rej_tasks	CPU
1	0.00	0.77	0.77	0	91.42
2	35.06	109.32	97.71	24	133.43
3	147.69	147.69	0.00	99	249.49
4	127.26	127.26	0.00	100	187.18
5	28.19	82.58	67.15	19	171.57

Table 5.3. Synthetic workloads with $E[B_j] = 150$

It is interesting to note that for increasing values of λ, unlike what we can expect, the average rejection cost (and the number of rejected tasks) does not always tend to increase due to an increase of the number of rejected tasks and the biggest value is never in correspondence to the biggest value of λ. See for example the set of instances with $E[B_j] = 150$ in which the smallest number of rejected tasks occurs for the extreme values of $\lambda = 1$ and $\lambda = 5$. For λ ranging from 1 to 5, avg_rej_cost ranges from 0 to 45.71 when $E[B_j] = 50$, from 0 to 94.97 when $E[B_j] = 100$, and from 0 to 147.69 when $E[B_j] = 150$. In general it is remarkable to note that, even if the number of rejected tasks has a large variance in the three cases, the average value is equal to 56 in the first case, 53 in the second and 48 in the third: it means that, according to an increasing rejection cost, the average number of rejected tasks decrease. Looking at the $avg\_tard\_cost(sched)$ values, it is worth noting that they are always lower than the average maximum tardiness penalty costs $E[B_j]$. Another fact to be noted is that for two instances the heuristic finds a solution in which all tasks are rejected and there is no $avg\_tard\_cost(sched)$ and for other two instances the number of rejected tasks is 99. This phenomenon may be justified by the incapability of followers to schedule all the tasks within their due-date so, considering the tardiness cost that would be generated, it is more convenient for the leader to reject all the tasks.

Finally, we note that the running times do not tend to increase in a constant way for increasing values of λ, but the lowest CPU time occurs always for $\lambda = 1$ and it seems that the biggest computational effort is for $\lambda = 2$ ($E[B_j] = 50$) and $\lambda = 3$ ($E[B_j] = 100$ and $E[B_j] = 150$). Even if the running times seem not to be limited for a heuristic algorithm, it is necessary to consider that the model we propose is inherently difficult to solve due to its discrete bilevel structure. Recall that DBLP is the hardest class of bilevel problems. In the literature there are very few applications of DBLP to real problems and no one is for big instances: indeed the application of bilevel models is still limited by the lack of efficient solution methods despite their capability to model problems in a more realistic fashion than the classical linear single level models.

Real workloads

Real workloads are taken from the Grid Workload Archive (GWA) developed by Iosup et al. [77] which is a big repository collecting real data from nine different Grid systems. Unfortunately, only a few workloads were complete and suitable for our model; for this reason we made the following assumptions. The task arrival date r_j was taken by the GWA instances used for the simulation. We considered the first $n = 100$ tasks arriving in the system. The computational resource units p_j^s required by task j on site s was provided by the GWA. The run_time was defined as for synthetic workloads (i.e., following the assumption that 2 out of the 3 available computational resources of a site are allocated to a task).

All the other data not provided by the GWA, i.e. task due-dates d_j, tardiness penalty costs w_j and task rejection costs B_j, were computed as for the synthetic workloads. In particular, we considered 5 real instances, called GWA_1, GWA_2, GWA_3, GWA_4 and GWA_5, and experimented the tabu search algorithm in three different scenarios, with $E[B_j] = 50, 100, 150$ and with the same parameters setting above used (i.e. $max\_iter = 15$, $max\_tabu = 5$, time limit 300 seconds and τ sufficiently large). The results are listed in Tables 5.4-5.6.

$inst$	avg_rej_cost	avg_tot_cost	avg_tard_cost	rej_tasks	CPU
GWA_1	49.45	49.45	0.00	98	168.58
GWA_2	39.79	40.00	1.18	83	122.72
GWA_3	30.38	30.38	0.00	64	149.97
GWA_4	42.39	42.59	0.00	94	127.27
GWA_5	39.87	39.87	0.00	83	127.27

Table 5.4. Real workloads with $E[B_j] = 50$

$inst$	avg_rej_cost	avg_tot_cost	avg_tard_cost	rej_tasks	CPU
GWA_1	96.08	96.08	0.00	96	159.26
GWA_2	63.31	80.62	52.45	67	125.50
GWA_3	61.73	61.73	0.00	64	150.21
GWA_4	65.87	65.87	0.00	68	105.23
GWA_5	43.00	44.55	2.77	44	125.49

Table 5.5. Real workloads with $E[B_j] = 100$

$inst$	avg_rej_cost	avg_tot_cost	avg_tard_cost	rej_tasks	CPU
GWA_1	145.98	145.98	0.00	98	170.57
GWA_2	95.09	116.06	63.55	67	125.35
GWA_3	91.88	91.88	0.00	64	125.99
GWA_4	80.65	82.42	3.93	55	191.58
GWA_5	121.19	121.19	0.00	83	126.86

Table 5.6. Real workloads with $E[B_j] = 150$

For the 5 real instances we compared the results for different value of the expected rejected cost $E[B_j]$. The aim is to make a sensitivity analysis with respect to the increase of the rejection cost. The first and most remarkable result is that according to what we can expect, the largest number of rejected tasks for every instances occurs always for $E[B_j] = 50$: it means that, when the rejection cost increases the leader tends to accept more tasks counterbalancing with an increase of the tardiness cost. Looking at the average value of the rejected tasks, we have that in the first case 84 tasks are rejected on average, 68 in the second and 73 in the third.

In this simulation there are no instances in which all the tasks are rejected, but the overall number of rejected tasks is quite large and bigger than what happens in the synthetic simulations. Moreover, it is important to note that, unlike the previous simulation in which the average tardiness cost was zero in 4 cases and for instances with 100 or 99 rejected tasks, in the real workloads simulation in 10 cases there is no tardiness cost, but in 6 cases the rejected tasks are less than 85. This result implies that, once the tasks are accepted, the followers are more able to schedule them without exceeding their due-dates. The combination of this two results may be explained by the following observation. The tasks require computational resource units p_j^s with a large variance, while the due-dates are computed on the basis of the expected value of the run time: two tasks, arriving in the system very close to each other, may have similar due-dates but very different p_j^s. For the leader it is more convenient to accept the smallest task and reject the other one with the final result that the number of rejected tasks is very large and the accepted tasks are the only ones that can be easily scheduled by the followers, generating no tardiness costs.

Finally, we note that the running times tend to be very similar no matter what is the value of the rejected cost except for instances GWA_4.

Summing up, Table 5.7 lists the average values of all the indicators used for the synthetic and the real scenarios.

$E[B_j]$	$scenario$	avg_rej_cost	avg_tot_cost	avg_tard_cost	rej_tasks	CPU
50	synth	25.31	32.17	13.04	56.01	114.45
	real	40.38	40.46	0.24	84.41	139.12
100	synth	49.31	62.39	26.91	52.80	135.47
	real	66.00	69.77	11.04	67.81	133.14
150	synth	67.64	93.52	33.13	48.40	166.62
	real	106.96	111.51	13.50	73.39	148.07

Table 5.7. Comparison between the two simulated scenarios

The running times are almost the same when $E[B_j] = 100$ and are comparable when $E[B_j] = 50$ (where the synthetic scenario is solved faster), and $E[B_j] = 150$ (where the performance in the real scenario is better). The analysis of the average values confirms what was previously observed: in the real scenarios there is a larger number of rejected tasks, i.e., rejected cost, and a smaller average tardiness cost with respect to the synthetic scenario. Despite this opposite behaviour, the average total costs are comparable and the performance of the algorithm is not affected by the use of real data rather than synthetic ones.

A comparison with respect to a simple heuristic and two lower bounds

In order to assess the quality of the solutions computed by the proposed tabu search algorithm, we compared the latter to a naive heuristic approach that works as follows: a feasible vector of leader's variables is randomly generated (i.e. the assignment of tasks to sites is randomly determined) and then the follower problem is solved by fixing the leader's variables found for computing a bilevel–feasible solution. This process is repeated iteratively within a fixed time limit equal to 300 seconds. In order to realize a fair comparison the random heuristic and the tabu search were run on the same instances with both synthetic and real workloads and in column *CPU* of the random heuristic in Tables 5.8-5.10 and Tables 5.11-5.13 we listed the elapsed time in seconds until the best solutions had been found.

Finally we solved two different relaxed versions of problem SLF for computing two lower bounds on the optimal solution. The first bound (denoted as *Bound_1*) was computed making the following relaxation: (i) we removed all the constraints concerning with the optimality conditions of the followers' problems (i.e., KKT conditions, primal and dual followers' problems feasibility), (ii) we relaxed the integrality constraints on the upper level variables. In the second lower bound (denoted as *Bound_2*) we also relaxed the integrality constraints on the followers' variables so obtaining a linear problem. The time limit imposed for the lower bounds calculated is 7200 seconds.

A first comparison was made on the instances with synthetic workloads.

λ	$Tabu\_Search$		$Random\_Heuristic$		$Bound\_1$	$Bound\_2$
	avg_tot_cost	CPU	avg_tot_cost	CPU	avg_tot_cost	avg_tot_cost
1	**11.15**	75.48	14.38	171.78	-	0.00
2	**40.04**	166.59	45.50	289.45	24.49	5.85
3	**42.04**	124.75	45.16	273.68	27.88	11.35
4	**21.92**	96.72	57.02	128.43	20.57	0.00
5	**45.71**	108.71	102.82	124.67	-	0.00

Table 5.8. Comparison for synthetic workloads with $E[B_j] = 50$

In Tables 5.8-5.10 we compared the average total cost of the solutions obtained by the two heuristics, i.e., the tabu search and the random heuristic, with the two bounds computed by solving two different relaxations. The dashes correspond to instances for which it was not possible to compute a feasible solution within the fixed time limit.

First of all it is important to note that the tabu search algorithm always achieves the better solution, which is listed in bold, and finds a feasible solution in less time than the random heuristic in all the instances tested but one ($\lambda = 1$ and $E[B_j] = 150$). Hence, the tabu search is clearly more effective and efficient than the random heuristic because it exploits the bilevel model. Looking at the two lower bounds, we can note that the second one

λ	$Tabu\_Search$		$Random\_Heuristic$		$Bound\_1$	$Bound\_2$
	avg_tot_cost	CPU	avg_tot_cost	CPU	avg_tot_cost	avg_tot_cost
1	**0.11**	55.39	10.84	84.88	-	0.00
2	**59.83**	124.33	96.18	269.15	56.03	26.54
3	**94.97**	248.65	121.41	248.78	55.17	9.13
4	**87.44**	124.35	95.00	205.13	63.28	24.16
5	**69.61**	124.60	96.84	211.27	68.86	30.11

Table 5.9. Comparison for synthetic workloads with $E[B_j] = 100$

λ	$Tabu\_Search$		$Random\_Heuristic$		$Bound\_1$	$Bound\_2$
	avg_tot_cost	CPU	avg_tot_cost	CPU	avg_tot_cost	avg_tot_cost
1	**0.77**	91.42	38.07	73.12	-	0.00
2	**109.32**	133.43	110.30	286.89	70.18	0.00
3	**147.69**	249.49	148.89	286.10	92.66	10.19
4	**127.26**	187.18	145.53	201.37	104.29	32.27
5	**82.58**	171.57	114.22	114.89	-	0.00

Table 5.10. Comparison for synthetic workloads with $E[B_j] = 150$

is less tight because it is computed by solving a strongly relaxed version of the problem: in more than half of the instances the average total cost is zero and the bound is not significant. On the other hand, the computation of the first bound requires a very large amount of time and in 6 out of 15 instances it was not possible to compute a solution within the time limit of 7200 seconds.

Comparing the value of the avg_tot_cost we can note that in the worst case the solution computed by the tabu search heuristic provide an avg_tot_cost which is almost the double of that provided by the first lower bound solution but in most cases the difference ranges from a minimum of about 1% to a maximum of about 55% with an average difference of about 27%.

The same comparison was realized on the instances with real workloads and listed in Tables 5.11-5.13.

$inst$	$Tabu\_Search$		$Random\_Heuristic$		$Bound\_1$	$Bound\_2$
	avg_tot_cost	CPU	avg_tot_cost	CPU	avg_tot_cost	avg_tot_cost
GWA_1	**49.45**	168.58	50.44	261.53	18.00	3.23
GWA_2	**40.00**	122.72	50.34	238.34	39.04	39.05
GWA_3	**30.38**	149.97	49.79	254.18	29.84	29.84
GWA_4	**42.59**	127.27	49.62	218.43	27.07	27.07
GWA_5	**39.87**	127.07	50.33	233.72	20.21	13.53

Table 5.11. Comparison for real workloads with $E[B_j] = 50$

Also in this case the tabu search algorithm clearly outperforms the random algorithm, both in term of solution

inst	*Tabu_Search*		*Random_Heuristic*		*Bound_1*	*Bound_2*
	avg_tot_cost	*CPU*	*avg_tot_cost*	*CPU*	*avg_tot_cost*	*avg_tot_cost*
GWA_1	**96.08**	159.26	99.61	264.18	31.42	6.97
GWA_2	**80.62**	125.50	99.40	232.27	70.37	62.16
GWA_3	**61.73**	150.21	100.04	261.94	60.63	60.63
GWA_4	**65.87**	105.23	99.54	255.14	64.99	64.99
GWA_5	**44.55**	125.49	100.36	221.57	41.06	41.02

Table 5.12. Comparison for real workloads with $E[B_j] = 100$

inst	*Tabu_Search*		*Random_Heuristic*		*Bound_1*	*Bound_2*
	avg_tot_cost	*CPU*	*avg_tot_cost*	*CPU*	*avg_tot_cost*	*avg_tot_cost*
GWA_1	**145.98**	170.57	149.05	253.32	44.66	9.45
GWA_2	**116.06**	125.35	151.21	268.12	106.45	93.50
GWA_3	**91.88**	125.99	149.51	237.21	90.11	90.11
GWA_4	**82.42**	191.58	149.13	272.37	79.54	79.54
GWA_5	**121.19**	126.86	147.12	223.19	58.65	38.83

Table 5.13. Comparison for real workloads with $E[B_j] = 150$

quality and computational time. On the one hand, the difference in terms of average total costs is smaller, but on the other hand the difference in terms of CPU times is more remarkable. Unlike the synthetic scenario, for every instance both lower bounds can be computed within their time limit. Note that the difference between *Bound_1* and *Bound _2* is less notable than the previous scenario suggesting that the two relaxing procedures implemented allow to find two valid and good lower bounds.

5.1.7 Conclusions

In the Grid scheduling problem considered in this section a set of independent tasks are submitted to an External Scheduler (ES) who is in charge to individuate the most profitable computing site for their execution. The allocation is driven by a cost function that takes into account both the tardiness cost of a task and the chance to reject such a task in case no profitable site is found i.e., the tardiness cost of the task would exceed a given maximum threshold. While the ES looks for executing the submitted tasks over the Grid minimizing the total cost for rejecting or delaying tasks, the goal of the Local Scheduler (LS) of each site is maximizing computational resource usage efficiency. The ES problem, together with that of each LS, form a hierarchical optimization problem, where the decision of the ES constraints and is constrained by that of the LSs. In fact, the tardiness cost values are dependent on the LSs' decisions.

The problem has been modelled as a DBLP and reformulated as a single level mixed-integer problem by introducing in the leader problem the KKT conditions of the followers problems. By testing the latter reformulation with CPLEX, we found out that the solver was able to solve the reformulation at the optimum within a reasonable computing time for the application only on instances with at most five tasks. This led us to design a heuristic algorithm to cope with large size instances happening in practice preserving and exploiting the

bilevel nature of the model.

The algorithm works in two phases: the first phase initializes the algorithm by computing a particular linear relaxation of the problem; the second phase works iteratively and uses a tabu search mechanism for finding a bilevel–feasible solution. Computational results are presented and discussed. The algorithm was tested using both synthetic workloads and real workloads and the results of both scenarios are consistent. The analysis of the results show that the algorithm is able to compute good solutions compared to two lower bounds within a reasonable computing time. This result is even more remarkable when real workloads are solved. Moreover, the algorithm clearly outperforms in terms of solution quality a competing naive heuristic algorithm which does not exploit the bilevel nature of the model. Future work may be focused on proper techniques to enhance the performance of the tabu search heuristic modifying the mechanism of the local search and on a more efficient and effective way to compute valid lower bounds in order to assess the quality of the solutions calculated by the heuristic algorithm.

5.2 A branch and cut based algorithm for a bilevel capacitated facility location problem

We address a particular capacitated facility location problem that is formulated as a discrete–continuous bilevel linear problem. In the proposed model, the upper level decision maker, denoted as the leader, sets a subset of facilities to open and the capacity of each facility; conversely, the lower level decision maker, call itdenoted as the follower, once the facilities have been designed, is in charge of satisfying the demand of a given set of clients beyond a certain threshold. The leader can foresee but can not control the follower's behaviour. The resulting formulation is a discrete–continuous bilevel problem. Although there exist in the literature exact methods for this class of bilevel problems, they can not be used to solve real size problems. We propose a branch and cut framework, in order to cope with the bilevel structure of the problem and the integrality of a subset of variables under control of the leader. The algorithm is exact in theory, but for computational reason we introduce suitable stopping criteria that make it a heuristic. A set of real benchmark instances available in the literature are used to test the algorithm.

5.2.1 Introduction

Facility location is a well known problem and has been widely investigated in the literature. Given a set of facilities with a fixed cost of activation and a set of demand sites to serve, the problem consists of setting the facility activation and the assignment of each demand site to an open facility in order to minimize the sum of the activation fixed costs and the variable assignment costs. This problem is also known as Uncapacitated Facility Location Problem (UFLP) and is one of the most studied. Karp [85] demonstrated UFLP is a NP-hard problem, while Krarup and Pruzan [90] showed the relation of UFLP to other combinatorial problems such as set-covering, set-packing, set-partitioning, p-median and p-center problems. Some of the first solution methods were proposed by Balinski and Wolfe [17], Efroymson and Ray [58], Khumawala [86], Cornuejols et al. [50], Erlenkotter [59]. A comprehensive survey on UFLP can be found in Cornuejols et al . [51].

A more realistic formulation of the problem considers a maximum capacity for each facility, i.e. once the facilities are opened the assignment problem is not trivial. The latter is known as Capacitated Facility Location Problem (CFLP) and it is much harder to tackle than the uncapacitated version. One of the first contribution on CFLP was provided by Kuehn and Hamburger [91] who proposed a heuristic algorithm designed for a warehouse location problem, and other solution methods were proposed, among the others, by Nauss [117], Akinc and

Khumawala [5] and Geoffrion and Graves [70]. An updated reviews of literature on facility location problems, in both the capacitated and uncapacitated version, is provided by Verter [144].

More recently a new branch of research is obtaining increasing attention as it aims at extending the classical facility location models taking into account the effect of uncertainties on the decision making process. Removing the hypothesis of deterministic parameters in the model, it is possible to define new solution strategies to cope with uncertain events, such as demand variation, cost estimation errors, facilities disruption or system failure. Stochastic optimization and robust optimization provide theoretical results necessary to solve these kind of problems under uncertainty. When there are no information about the probabilistic distribution of uncertain parameters, the most common approach used in the literature is to define particular robustness measures, such as the min-max cost or the min-max regret. The latter are well known worst case optimization methods and there is a wide literature on this kind of problems, see for instance Averbakh [16]. Snyder [138] presents an interesting and comprehensive survey of the more recent literature on stochastic and robust facility location models along with references to applications of facility location problems under uncertainty.

In the literature there are several examples of worst case analysis developed by means of bilevel programming. Scaparra and Church [128] [129] propose a bilevel model for minimizing the effect of disruption and interdiction of a set of facilities. In Arroyo and Galiana [10] the bilevel approach is used for the terroristic threat problem. Aksen and Aras [8] develop a matheuristic for the facility location and interdiction problem formulated as a static Stackelberg game and Kochetov et al. [88] propose a similar approach for the facility location problem used to model the competition of two decision makers on the same market.

In this section we investigate a CFLP formulated as a discrete–continuous bilevel linear problem. The proposed model originates from the stochastic facility location model proposed by Louveaux and Peters [101]. The authors utilize a scenario based approach to handle the uncertainty of demand and assignment profit and, unlike classic uncapacitated facility location models, assume that the size of each facility can be determined by the decision maker.

According to this framework, we propose a deterministic model but we assume the presence of two decision makers with an explicit hierarchical relation, a leader and a follower. The leader (e.g. an Authority) locates the facilities and defines their size, the follower (e.g. a private company) is in charge of satisfying the clients' demand and aims at maximizing his profit. The key assumption of the model is that the follower is not obliged to satisfy all the demands and the leader can not control, force or apply any sanction on him. The leader can only open more facilities or install more capacity in order to make profitable for the follower to satisfy all the demands beyond a given threshold fixed by the leader. Hence the two decision makers aim at different goals: the leader pursues both an economical and a social objective, while the follower pursues only an economical one. The difference of the objectives can be naturally represented by means of a bilevel model. In this section we describe the mathematical model and propose a branch and cut based algorithm that is tested on a set of real life benchmark instances. The rationale of the algorithm is the following: we solve the single level relaxation of the model to compute a lower bound, then the upper level variables are set and the optimal solution of the follower is computed. If this solution satisfies the leader's constraint, i.e. each client is served beyond a certain threshold, the latter is a feasible solution and provides an upper bound for the problem. Otherwise, given the set of open facilities, we check if it possible to compute a feasible solution if the leader increases the installed capacity. After this check, the solution is removed by means of a valid inequality and the algorithm iterates computing a new solution of the single level relaxation. The algorithm is able to find the optimal solution of the problem, in theory. Notwithstanding, we introduce suitable stopping criteria to limit the computational time and obtain more meaningful results when real life instances are solved.

The main contribution of this application is to provide a novel approach for discrete location problems. The framework we propose is typical of problems in which a player designs a system and a different player is assigned to manage it. To our knowledge, similar applications of bilevel programming have not been presented yet in the field of facility location problems.

5.2.2 Bilevel model formulation

The CFLP we address is formulated by means of bilevel programming. In the literature most of the real life applications involving bilevel programming consider mathematical models with continuous variables. We propose a DCBLP with mixed-integer upper level and continuous lower level variables. Both the objective functions and the constraints are linear.

Given a set I of n clients and a set J of m facilities, the leader has to open a subset of facilities and define their capacities with the aim of minimizing the total investment cost. Let f_j be the fixed cost for opening a new facility j and let g_j be the cost for each unity of capacity installed in the facility j. We assume that the capacity cost g_j is equal for all the facilities, i.e. $g_j = g \quad \forall j$. Once the facilities have been designed and opened, the follower is responsible for serving the clients. Let d_i be the demand for each client i, c_{ij} the shipment cost for serving client i from facility j, and p_i the price paid by client i for being served. If the follower assigns client i to facility j he gains a unitary profit $\pi_{ij} = p_i - c_{ij}$. The key assumption of the model is that the leader can not control the follower and can not impose him to serve all clients, thus the follower has no operative constraints and can manage the shipment service according to the maximization of his profit. Notwithstanding, in order to obtain an equitable service, the leader has to guarantee that each client is served beyond a certain threshold λ that is assumed to be the same for all clients. Let us consider, for instance, the problem of opening a set of landfill plants: the leader is a public Authority that has to locate and design the landfill plants, but the waste collection is managed by a private company. The Authority can not impose the company to serve all clients, he can only properly design the system so that it is profitable for the company to collect a percentage of waste beyond a certain threshold from each client.

The bilevel facility location problem $BFLP$ is formulated in the following as a DCBLP.

$$
\begin{aligned}
(BFLP) \quad & \min_{x,z,y} \sum_{j\in J} f_j\, x_j + \sum_{j\in J} g\, z_j && && (5.52)\\
& \sum_{j\in J} y_{ij} \geq \lambda && \forall i \in I && (5.53)\\
& x_j \in \{0,1\} && \forall j \in J && (5.54)\\
& z_j \geq 0 && \forall j \in J && (5.55)\\
& y \in \underset{y}{\operatorname{argmax}} \sum_{i\in I}\sum_{j\in J} d_i\,(p_i - c_{ij})\, y_{ij} && && (5.56)\\
& \sum_{j\in J} y_{ij} \leq 1 && \forall i \in I && (5.57)\\
& \sum_{i\in I} d_i\, y_{ij} \leq z_j && \forall j \in J && (5.58)\\
& y_{ij} \leq x_j && \forall i \in I,\, \forall j \in J && (5.59)
\end{aligned}
$$

$$y_{ij} \geq 0 \qquad \forall i \in I, \forall j \in J \qquad (5.60)$$

The leader's variables are x_j and z_j: according to (5.54) and (5.55), x_j is binary and is equal to 1 if the facility j is open and to 0 otherwise, and z_j is positive and represents the installed capacity of facility j. The follower's variable is y_{ij} and represents the percentage of demand of client i satisfied by facility j. The leader's objective function (5.52) is the minimization of fixed opening costs and variable capacity costs. Constraints (5.53), let us call them *social equality* constraints, express the social goal of the leader and impose that the optimal solution of the follower's problem (5.56)-(5.60) is such that each client's demand is satisfied beyond a certain threshold λ. The follower's objective function (5.56) is the maximization of the profit coming from the assignment of clients to facilities. Constraints (5.57) assure that client's demand is not exceeded, constraints (5.58) and (5.59) forbid the follower to assign a client to a facility if the facility is close or if there is no installed capacity.

From the bilevel perspective, y_{ij} is required to be a rational solution. If a solution y_{ij} is both rational and satisfies the leader's constraints (5.53), it is bilevel-feasible. Constraints (5.53), (5.58) and (5.59) show how the two decision makers affect each other: given a set of open facilities, i.e. after setting x_j and z_j, the follower can only use this set of facilities and their capacities for the assignment of demand, but if the follower does not provide an equitable solution, i.e. constraints (5.53) are violated, the leader is obliged to modify his solution.

A general and credible assumption we made is that there is a form of semi-cooperation between the two decision makers, thus the optimistic approach is adopted. It meas that if the follower has multiple optimal solutions and at least one satisfies the social equality constraint, the follower chooses the latter according to the leader's interest. Note that this assumption does not imply that the leader can control the follower in general.

In the following we define other two formulations that will be used in the algorithm described in the next section. The single level relaxation SLR of $BFLP$ is obtained by dropping the follower's objective function (5.56) as follows.

$$(SLR) \quad \min_{x,z,y} \sum_{j \in J} f_j \, x_j + \sum_{j \in J} g \, z_j \qquad (5.61)$$

$$\sum_{j \in J} y_{ij} \geq \lambda \qquad \forall i \in I \qquad (5.62)$$

$$x_j \in \{0,1\} \qquad \forall j \in J \qquad (5.63)$$

$$z_j \geq 0 \qquad \forall j \in J \qquad (5.64)$$

$$\sum_{j \in J} y_{ij} \leq 1 \qquad \forall i \in I \qquad (5.65)$$

$$\sum_{i \in I} d_i \, y_{ij} \leq z_j \qquad \forall j \in J \qquad (5.66)$$

$$y_{ij} \leq x_j \qquad \forall i \in I, \forall j \in J \qquad (5.67)$$

$$y_{ij} \geq 0 \qquad \forall i \in I, \forall j \in J \qquad (5.68)$$

Recall that the optimal solution of SLR provides a valid lower bound for $BFLP$.

Finally we define the bilevel problem obtained by the original $BFLP$ given a fixed set of open facilities, i.e. with $x_j = \bar{x}_j$. Let us call this problem, that is parameterized by $\bar{x}_j$, slave problem $SVP(\bar{x}_j)$.

$$(SVP(\bar{x}_j)) \quad \min_{z,y} \sum_{j\in J} g\, z_j \tag{5.69}$$

$$\sum_{j\in J} y_{ij} \geq \lambda \qquad \forall i \in I \tag{5.70}$$

$$z_j \geq 0 \qquad \forall j \in J \tag{5.71}$$

$$y \in \operatorname*{argmax}_{y} \sum_{i\in I}\sum_{j\in J} d_i\,(p_i - c_{ij})\, y_{ij} \tag{5.72}$$

$$\sum_{j\in J} y_{ij} \leq 1 \qquad \forall i \in I \tag{5.73}$$

$$\sum_{i\in I} d_i\, y_{ij} \leq z_j \qquad \forall j \in J \tag{5.74}$$

$$y_{ij} \leq \bar{x}_j \qquad \forall i \in I,\, \forall j \in J \tag{5.75}$$

$$y_{ij} \geq 0 \qquad \forall i \in I,\, \forall j \in J \tag{5.76}$$

In $SVP(\bar{x}_j)$ the leader can only modify the capacity of the open facilities. Note that, despite there is not an explicit relation between $\bar{x}_j$ and z_j, it is not convenient for the leader to increase z_j when $\bar{x}_j = 0$ because this does not affect the follower best response due to constraints (5.75).

In the next section we investigate how the mathematical models described are used to design a heuristic algorithm for solving $BFLP$.

5.2.3 General framework of the branch and cut based heuristic

We have already seen that one of the most common method used to solve bilevel problem with a linear follower problem, is to replace the latter by its KKT conditions. Despite its easy implementation, this approach can not be efficiently used to solve large size instances. Moreover, in $BFLP$ a subset of variables under control of the leader is binary and this substantially increases the computational burden required to solve the single level reformulation.

To cope with this criticality, we propose a heuristic approach that is designed as follows.

Step 0 Set lower bound lb = 0, incumbent solution $best$ equal to a sufficiently large number, max_iter and gap.

Step 1 Solve SLR, if it is infeasible STOP, otherwise update lb. If $lb \geq best$ STOP (the incumbent is optimal), else go to Step 2.

Step 2 (Bilevel-feasibility check) Solve the follower problem of $BFLP$ given the solution $(\bar{x}_j,\bar{z}_j)$ computed at Step 1. Let y^*_{ij} be the optimal solution of the follower: if $\sum_{j\in J} y^*_{ij} \geq \lambda \quad \forall i \in I$, update $best$ then STOP (the rational solution is also bilevel-feasible), else go to Step 3.

Step 3 (Slave problem resolution) Solve the slave model $SVP(\bar{x}_j)$ given the vector $\bar{x}_j$ computed at Step 1. Let $(\bar{x}_j, \hat{z}_j, \hat{y}_{ij})$ be the optimal solution and let $opt_{svp} = \sum_{j\in J}(g\,\hat{z}_j + f_j\,\bar{x}_j)$. If there is some $\hat{z}_j = 0$ update $opt_{svp} = opt_{svp} - \sum_{j|\hat{z}_j=0} f_j$. If $opt_{svp} < best$ then update $best$ and go to Step 3.1, else go to Step 3.2.

Step 3.1 If $\frac{best-lb}{lb} \leq gap$ then STOP.

Step 3.2 If max_iter iterations have been computed then STOP, else go to Step 4.

Step 4 (Cut generation) Given the vector $\bar{x}_j$ computed at Step 1, let $S^1 = \{j \mid \bar{x}_j = 1\}$ and $S^0 = \{j \mid \bar{x}_j = 0\}$, apply the following cut to SLR: $\sum_{j \in S^1} x_j + \sum_{j \in S^0} (1 - x_j) \leq |J| - 1$. If there is some $\hat{z}_j = 0$ update $S^1 = S^1 \setminus \{j\}$ and $S^0 = S^0 \cup \{j\}$ $\forall j$ such that $\hat{z}_j = 0$ and apply the additional cut $\sum_{j \in S^1} x_j + \sum_{j \in S^0} (1 - x_j) \leq |J| - 1$. Go to Step 1.

In Figure 5.3 a flow chart of the algorithm is depicted.

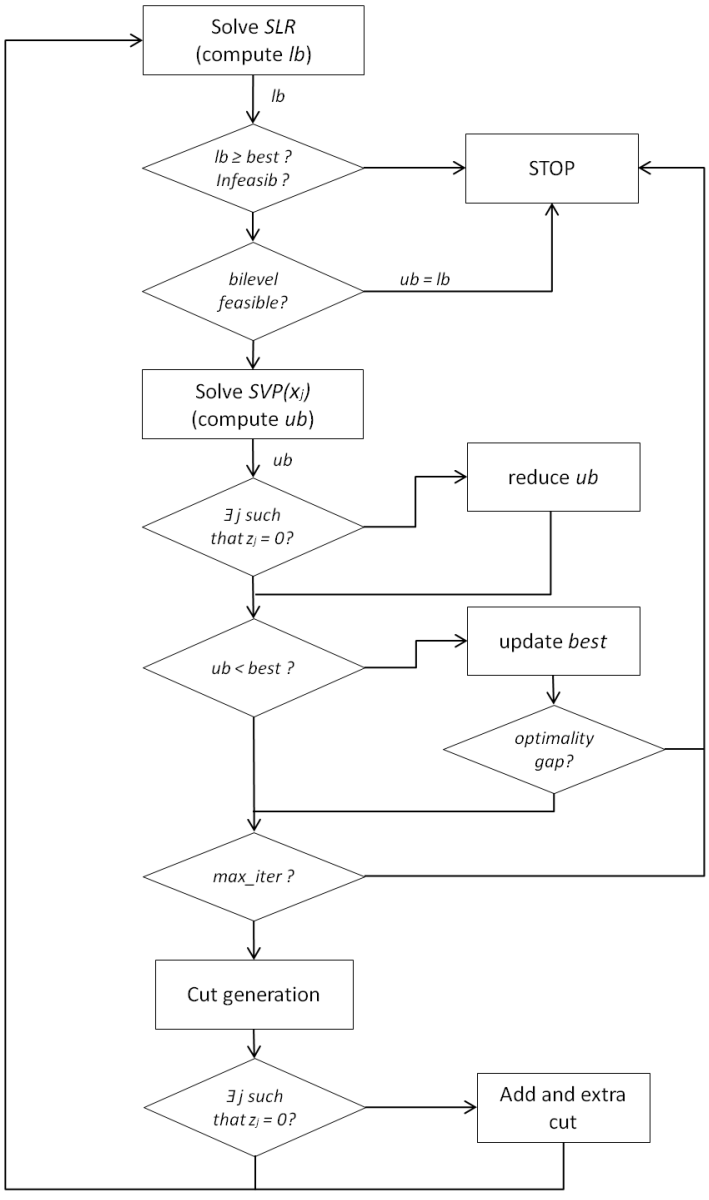


Figure 5.3. A flow chart of the branch and cut algorithm

The algorithm starts solving SLR to compute a lower bound, let us call the solution $(\bar{x}_j, \bar{z}_j, \bar{y}_{ij})$ (Step 1). Note that this lower bound may be greater than the incumbent solution, since every time a feasible solution of $BFLP$ is computed it is discarded by a cutting procedure (Step 4). Thus, if $lb \geq best$, the optimal solution has already been found and cut off, hence the algorithm can halt. Otherwise, we set the upper level variables at $(\bar{x}_j, \bar{z}_j)$ and solve the follower problem to compute the follower's reaction; let y^*_{ij} be the follower's optimal solution (Step 2). If y^*_{ij} satisfies the social equality constraints (5.53), $(\bar{x}_j, \bar{z}_j, y^*_{ij})$ is bilevel-feasible and provides an upper bound for $BFLP$ (Step 2). In this case, as the follower's variables y_{ij} do not affect the leader's objective function, the value of the upper bound computed at Step 1 and the value of the lower bound computed at Step 2 coincide, hence $(\bar{x}_j, \bar{z}_j, y^*_{ij})$ is the optimal solution and the algorithm halts. Otherwise, y^*_{ij} is rational, but not bilevel-feasible.

It may be possible for the leader to increase the capacity of the open facilities, compute a new solution $(\bar{x}_j, \hat{z}_j, \hat{y}_{ij})$ and check for bilevel-feasibility: this coincides with solving slave problem $SVP(\bar{x}_j)$ (Step 3). The solution computed solving $SVP(\bar{x}_j)$ is the best for the leader given a set of open facilities. $SVP(\bar{x}_j)$ is a bilevel model like $BLFP$, but unlike the latter, binary variables x_j are fixed so the model is a BLP. Notwithstanding, its resolution still remains hard from a computational point of view. In order to circumvent this criticality, in the following section we investigate the slave problem more in detail and propose an efficient branch and bound method properly designed. After solving $SVP(\bar{x}_j)$ there may exist open facilities in which the installed capacity is zero; the vector $\bar{x}_j$ is updated to $\hat{x}_j$ just closing the unused facilities, the opening costs for these facilities are discounted and a new feasible solution is computed.

Finally any solution with $x_j = \bar{x}_j$ (and in case $x_j = \hat{x}_j$) is discarded and the algorithm iterates from scratch. The cut used is

$$\sum_{j \in S^1} x_j + \sum_{j \in S^0} (1 - x_j) \leq |J| - 1$$

where $S^1 = \{j \mid \bar{x}_j = 1\}$ and $S^0 = \{j \mid \bar{x}_j = 0\}$. The algorithm iterates until at least one of the following conditions occur: (i) SLR is infeasible, (ii) the optimal solution of SLR is worse than the incumbent, (iii) the optimality gap is within a certain threshold, (iv) a maximum number of iterations is reached. If stopping criterion (i) or (ii) occur, it means that the cuts added to SLR make it infeasible or cut off the optimal solution: in both cases the optimal solution has already been computed in the previous iterations and the incumbent is the optimum. Note that, without stopping criterion (iv) and setting $gap = 0$, the algorithm computes the optimal solution of $BFLP$. Indeed, for each possible subset S^1 and S^0, a bilevel-feasible solution is computed and after that it is discarded. Hence, in the worst case, all the possible bilevel-feasible solutions are computed and the optimum is found. For the sake of algorithm application, we introduced a positive optimality gap and a maximum number of iterations for limiting the computational time.

Let us consider a network $G = (I, J)$, with a node for each client i and for each facility j, an arc (i, j) of cost c_{ij}, that represents the assignment cost for serving client i from facility j and a positive weight p_i for each node i, that represents the price paid by client i for being served. Let us assume a client, say client i^*, is assigned to an open facility j for which profit $\pi_{i^*j} = (p_{i^*} - c_{i^*j}) < 0$ and that all the other facilities k for which profit $\pi_{i^*k} > 0$ are closed. It follows that the social equality constraints $\sum_{j \in J} y_{i^*j} \geq \lambda$ are never satisfied, because the follower gains a negative profit from assigning client i^* to facility j. According to this optimality requirement for solution y_{ij}, we can modify the SLR by adding the following constraints

$$y_{ij} = 0 \qquad \forall i \in I, \forall j \in J \quad \text{such that} \quad (p_i - c_{ij}) < 0 \tag{5.77}$$

Note that arcs (i, j) with zero profit (i.e. $\pi_{ij} = 0$) are not excluded for the semi-cooperative assumption:

if an optimal follower's solution does not satisfy the social equality constraints and it is possible to restore bilevel-feasibility using zero profit arcs, the follower objective function does not change, but the new solution is bilevel-feasible and it is preferred by the follower.

It follows that if SLR admits a feasible solution $(\bar{x}_j, \bar{z}_{ij})$, all the arcs from the open facilities $\bar{x}_j = 1$ must have non negative profit.

Proposition 16. *If SLR is solved adding constraint (5.77) and the solution at Step 2 is not bilevel-feasible, it is possible to modify the installed capacity z_j, so that the optimal follower's solution also satisfies the social equality constraints.*

Proof: The solution computed solving SLR with constraint (5.77) is such that for each open facility j and each client i there exists at least one arc (i, j) with non negative profit. As the objective of the follower is to maximize the profit, given a subset of open facilities, the higher is the installed capacity, the more demand will be served by the follower. In a trivial case, it is possible to install in each open facility a capacity equal to the sum of all the client's demand. □

Consider the example in Figure 5.4 in which $\lambda = 0.5$, three facilities are open and the installed capacity is 2.

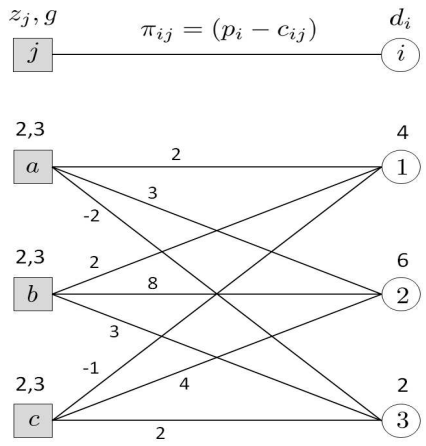


Figure 5.4. An example

Although the installed capacity is adequate to cover at least the 50% of each client's demand, the follower gains the maximum profit of 30 just serving client 2. A trivial possibility for the leader is to increase the capacity to $z_j = \sum_i d_i$, i.e. $z_a = z_b = z_c = 10$ with a capacity cost of 90. Otherwise, a better solution can be achieved computing the minimum amount of installed capacity z_j such that the follower's optimal solution satisfies all the social equality constraints. The latter is exactly $SVP(x_j)$ whose optimal solution is $z_a = 1$, $z_b = 6$ and $z_c = 1$ with a capacity cost of 24. Note that, regardless the capacity cost, in any bilevel-feasible solution all the demand of client 2 has to be satisfied since $\pi_{2j} \geq \pi_{1j}$ and $\pi_{2j} \geq \pi_{3j}$ for $j = a, b, c$.

In the next section we present a branch and cut algorithm to efficiently solve $SVP(x_j)$. This algorithm, denoted as *Slave Algorithm*, is used like a subroutine in the general branch and cut algorithm we propose.

5.2.4 Slave Algorithm for problem $SVP(x_j)$

The objective of solving $SVP(x_j)$ is to determine the capacity to be installed such that a ration solution is also bilevel-feasible. In this section we present a branch and bound algorithm to solve $SVP(x_j)$. Every node of

the branching tree is a possible assignment of a client i to a facility j, denoted with node $(i-j)$. The main rationale of the algorithm is to consider all the possible assignments and exploit some properties that allow to easily compute a lower bound for each branching node. Once the bound is computed, the node can be pruned or other subproblems can be generated according to the classical fathoming rules. Since in the worst case every client can be assigned to every facility, the maximum number of nodes is $\sum_{i=0}^{n} m^i = \frac{m^{n+1}-1}{m-1}$.

In the following we describe in detail how the algorithm works and apply it on a small example.

Priority list $\mathcal{L}$

For each possible assignment $i-j$ such that $\pi_{ij} \geq 0$, we define a priority list $\mathcal{L}_{ij}$ which is a row vector of n elements (the number of clients). $\mathcal{L}_{ij}$ is computed as follows:

1. given facility j, consider all the other clients k
2. if $\pi_{ij} < \pi_{kj}$ set the k^{th}-element of $\mathcal{L}_{ij}$ equal to 1 and 0 otherwise.

Furthermore, we define δ_{ij} as the number of non-zero elements of $\mathcal{L}_{ij}$. Given a possible assignment $(i-j)$, in $\mathcal{L}_{ij}$ there is 1 for every client that "dominates" client i in terms of profit from the follower's perspective. It implies that if a capacity is installed in j to serve i this capacity is used by the follower to serve all the other dominating clients.

Let us recall the example in Figure 5.4. The priority list for each client is:

$\mathcal{L}_{ij}$	1	2	3	δ_{ij}
$\mathcal{L}_{1a}$	0	1	0	1
$\mathcal{L}_{1b}$	0	1	1	2
$\mathcal{L}_{2a}$	0	0	0	0
$\mathcal{L}_{2b}$	0	0	0	0
$\mathcal{L}_{2c}$	0	0	0	0
$\mathcal{L}_{3b}$	0	1	0	1
$\mathcal{L}_{3c}$	0	1	0	1

From the priority list it is clear, for instance, that client 1 can not be served by facility b, unless clients 2 and 3 are fully satisfied. If the demand of a client i is totally covered, we define i a *filled* client, otherwise it is *unfilled*.

Remark 1. *If there exists an assignment $(i-j)$ such that $\delta_{ij} = 0$ for each client, the optimal solution of SVP(x_j) is trivial and every client is assigned to the facility such that it is not "dominated" by other clients. The minimum capacity to be installed is $\sum_j z_j = \lambda \cdot \sum_i d_i$.*

Note that the assumption $g_j = g \quad \forall j$ is fundamental to compute the optimal solution through the priority list: indeed, it is not relevant in which facility the capacity is installed, but the optimal solution depends only on the total amount.

Lower bound computation

Given a node $(i-j)$ of the branch and bound tree, it represents a partial assignment of clients. It is possible to compute a lower bound for node $(i-j)$ updating its associated priority list as follows:

$$\mathcal{L}'_{ij} = \{\mathcal{L}_{ij} \cup \mathcal{L}_{kv}\} \qquad \forall \text{ node } (k-v) \text{ predecessor of node } (i-j)$$

We denote the q^{th}-element of $\mathcal{L}'_{ij}$ with $\mathcal{L}'_{ij}[q]$. The lower bound at node $(i-j)$ is

$$LB_{ij} = \sum_{q \in I \mid \mathcal{L}'_{ij}[q]=1} d_q + \sum_{q \in I \mid \mathcal{L}'_{ij}[q]=0} \lambda\, d_q$$

If $\mathcal{L}'_{ij}[q] = 1$, client q is filled and the whole demand d_q is considered, otherwise we consider only the percentage $\lambda\, d_q$.

We prove the following results.

Theorem 18. *Given a node $(i-j)$ and the associated partial assignment, the capacity to be installed in order to guarantee bilevel-feasibility is greater than or equal to LB_{ij}.*

Proof: The lower bound LB_{ij} is computed taking into account the priority list of node $(i-j)$ and its predecessors. We show that it is not possible to install a capacity smaller than LB_{ij} unless the assignment changes or does not satisfy the social equality constraints. If client i is unfilled and served by j, we can not reduce z_j otherwise the demand satisfied is beyond threshold λ or the missing demand is covered by another facility, that is the assignment changes. If client i is filled and we reduce z_j, the solution is still feasible (the covered demand is greater than $\lambda\, d_i$), but from the priority list it occurs that there exists a client k assigned to a facility v such that $\pi_{iv} > \pi_{kv}$, hence the follower gains a higher profit changing the assignment. Hence the proof. □

Remark 2. *There always exists a trivial feasible solution in which all clients but one are filled. Let k be the client with the minimum demand. All the solutions of SVP(x_j) with $\sum_{j \in J} z_j > \sum_{i \in I, i \neq k} d_i + \lambda\, d_k$ are not rational. Hence, if a node $(i-j)$ has a priority list $\mathcal{L}_{ij}$ with all 1, the node can be pruned.*

The clients are sorted for the assignment according to the following branching rule: we choose the client with the smallest number of non-zero elements in its priority list and we branch the node with the smallest lower bound. Recalling the priority list reported above, clients are sorted and selected in the following order: $2-3-1$.

Upper bound computation: rationality test

Unlike a classical branch and bound algorithm, once all clients have been assigned, the lower bound computed at a leaf of the branching tree does not necessarily provide a valid upper bound. In this subsection we show how to compute an upper bound once all clients have been assigned.

Let us consider the example in Figure 5.4. In Figure 5.5 a partial branch and bound tree is represented until the first leaf is generated.

Starting from the root node, whose lower bound is 5, client 2 is selected and nodes $(2-a)$, $(2-b)$ and $(2-c)$ are generated. The algorithm continues branching on node $(2-a)$. Client 3 is selected and other 2 nodes are generated and so on. Finally at node $(1-a)$ all clients are assigned and the path from the root to the leaf $(1-a)$ may represent a follower's rational solution. $\mathcal{L}_{1a} = \{0, 1, 0\}$, client 2 is filled, while 1 and 3 are unfilled. Clients 1 and 2 are assigned to a and 3 is assigned to b, $z_a = 7$, $z_b = 1$ and $z_c = 0$: the total installed capacity is 8, the total cost is 24 and the profit gained by the follower is 22. Actually, if we solve the follower problem, given the same installed capacity, the maximum profit is 27 and the rational solution is $y_{1a} = 0.5$, $y_{2a} = 0.83$, $y_{2b} = 0.17$, that is 1 is unfilled and assigned to a, 2 is filled and assigned to both a and b and 3 is not assigned at all. Thus,

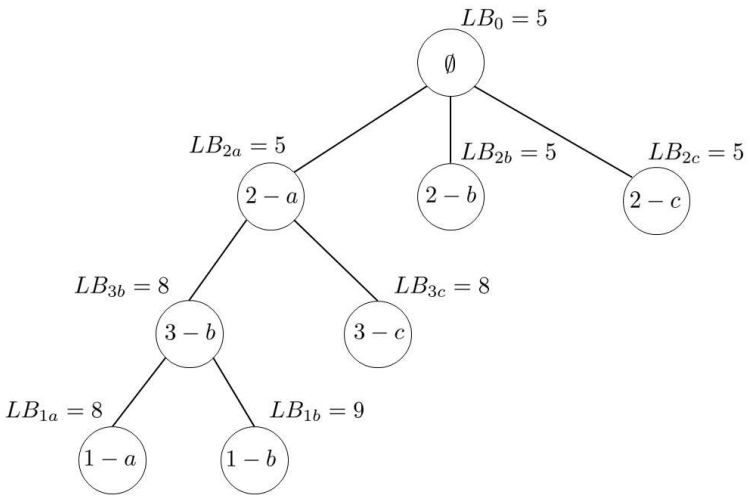


Figure 5.5. Partial branch and bound tree

the solution corresponding to path $(2-a)$, $(3-b)$ and $(1-a)$ does not provide a valid upper bound as it is not a rational solution.

This drawback occurs because $SVP(x_j)$ is a bilevel problem and the solution computed by the branch and bound does not completely take into account the reaction of the follower. The priority list tries to consider the follower's best response, but it may not be enough.

In order to assess if a solution is rational it is possible to easily perform a rationality test. We solve a shortest path problem on an auxiliary network built as follows:

1. consider only assignments $i-j$ such that $\pi_{ij} \geq 0$
2. introduce two dummy nodes, a source s and a sink t
3. introduce a directed arc (s,i) of cost 0 from s to every client i
4. introduce a directed arc (i,t) of cost 0 from every *unfilled* client i to sink t
5. if a client i is assigned to a facility j, introduce a directed arc (i,j) of cost $-\pi_{ij}$
6. for every *unfilled* client i introduce a directed arc (j,i) of cost π_{ij}, $\forall j$ such that $z_j > 0$
7. for every *filled* client i introduce a directed arc (j,i) of cost π_{ij}, $\forall j$ such that $z_j > 0$ and i is not assigned to j.

See Figure 5.6 for a representation of the auxiliary network associated to solution $(2-a)$, $(3-b)$ and $(1-a)$. Let us give a flavour of the rationale used to build the network. A $s-t$ path represents a redefinition of the assignment inside the network. For the sake of clearness, let us consider for instance the path $s-2-a-1-t$ of cost -1. If one unit of demand used to serve client 2 is differently assigned to serve client 1, this produces a marginal profit of -1, thus the follower has not interest in changing the assignment. Otherwise, if we consider the path $s-3-b-2-a-1-t$, the cost is 4, it means that changing the assignment this provides a marginal profit of 4. Indeed, the follower never chooses the assignment $(2-a)$, $(3-b)$ and $(1-a)$ because, with the same amount of installed capacity, there exists another solution which provides a largest profit, thus the solution is not rational.

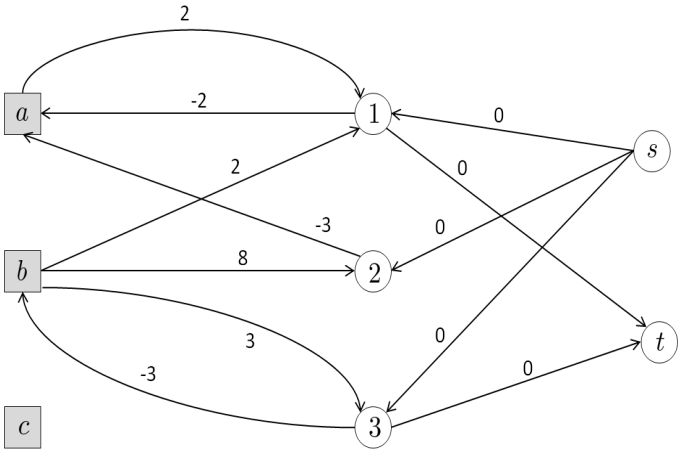


Figure 5.6. Auxiliary network associated to solution $(2-a)$, $(3-b)$ and $(1-a)$

Given an assignment, the rules used to build the network take into account all the possible changes in the assignment. If client i is unfilled and it is served by facility j, the latter can increase or reduce the served demand, hence there is an arc (i,j) and an arc (j,i) of cost $-\pi_{ij}$ and π_{ij}, respectively (rules 5 and 6). Of course client i can be also assigned to another facility j with $z_j > 0$ yielding a profit increase of π_{ij} (rule 6). Otherwise, if client i is filled and it is served by facility j, the latter can only reduce the served demand, thus there is only an arc (j,i) (rule 7). Any extra demand can be served from another facility v only if the demand already served by j is reduced of an equivalent amount toward another client of the network. For this reason there are not arcs (i,t) (rule 4) from filled clients to the sink node.

Theorem 19. *Given a solution corresponding to a leaf in the branching tree, the solution is rational if and only if the longest $s-t$ path in the associated auxiliary network has length equal to 0.*

Proof: (*Necessary*) Let us assume, by contradiction, that there exists one $s-t$ path with a positive length in the auxiliary network. It means that, once the capacity is installed, the follower has the possibility to change the assignment and earn an additional profit. Hence, the assignment does not correspond to the best follower's response.
(*Sufficient*) If there are no paths of positive length, every changing in the assignment does not change the profit or yields a profit reduction for the follower. Hence he has not interest in modifying the solution. □

A solution which satisfies the rationality test is rational and also bilevel-feasible as it is build to satisfy the social equality constraints. Note that, if the longest path in the auxiliary network is equal to 0, the follower has multiple optimal assignments, but, for the semi-cooperative assumption, he chooses the rational solution provided by the branch and bound algorithm as it satisfies the social equality constraints.

Finally, the algorithm does not consider the assignment of a client to two or more facilities according to the following result.

Theorem 20. *In the described branch and bound scheme, given a rational solution with multiple assignments of a client to facilities, it is always possible to compute a rational solution with single assignment for each client.*

Proof: Let us assume to assign a client i to facilities j and v and let us perform a rationality test. The capacity used to serve client i is installed on facilities j and v. We can move all the capacity used to serve i from j to v, if

$\pi_{ij} \leq \pi_{iv}$, and from v to j, otherwise, obtaining a single assignment for client i. The auxiliary network of this new solution has less arcs and the same installed capacity. Hence, it is possible to compute a single assignment solution with the same capacity whose auxiliary network has a smaller number of arcs, i.e. a smaller number of $s-t$ paths. If the multiple assignment solution is rational, also the corresponding single assignment solution is rational, while the contrary is not true. Hence the proof. □

Summing up all the theoretical results, the following corollary holds.

Corollary 3. *The Slave Algorithm solves to optimality problem SVP(x_j).*

Proof: The algorithm is a branch and bound which considers all the possible assignments of clients to facilities. For Theorems 18, 19 and 20, the method is exact. □

Computational complexity of the rationality test

The rationality test can be easily performed solving a shortest path problem on the auxiliary network in which the arc costs are changed of sign. All the operations to build the network have a complexity of $O(n)$: it is sufficient to control whether a client is filled or not and properly add arcs according to rules 1-7. The Bellman-Ford algorithm for shortest $s-t$ path is used, see Ahuja et al. [1]; this algorithm is able to compute the shortest path with arbitrary arc costs and detects negative cycles. The algorithm has a computational complexity $O(|N||E|)$, where $|N|$ is the number of nodes and $|E|$ is the number of arcs. In the worst case in which all the nodes are unfilled, $|N| = n + m + 2$ and then we have:

- two arcs between a client and its assigned facility = $2n$
- an arc between each client and all the other facilities = $n(m-1)$
- arcs from s and to t = $2n$.

Thus $|E| = 4n + n(m-1)$. The complexity of the algorithm is $O(n^2m + nm^2)$. If $n = m$, as in almost all the benchmark problems present in the literature, it becomes $O(n^3)$. Alternatively, in order to verify the rationality of the solution, it may be possible to solve the follower problem of $SVP(x_j)$ with a commercial solver, as it is a linear programming problem, and then check if the social equality constraints are satisfied. It is valuable to note that in the worst case, if the follower problem admits multiple rational solutions, some of them violating the social equality constraints and some other not, it is necessary to solve the linear programming problem twice, unlike what happen solving the shortest path problem. Computational tests show that the overall performance of the shortest path approach is better than solving the follower problem by means of a commercial solver, mainly because there is a high number of multiple rational solutions in real life benchmark problems. Finally, it is useful to observe that the rationality test is applied given a subset of open facilities. Since the objective of the leader is to minimize the total cost, especially in the first iterations of the branch and cut algorithm, the subset of open facilities is very small. It follows that $m << n$ and the complexity can be approximated as $O(n^2)$.

An example

Let us recall example in Figure 5.4 in which all the facilities are open. As above noted, the rationality test of assignment $(2-a)$, $(3-b)$ and $(1-a)$ shows that the solution is not rational. The complete branch and bound tree is reported in Figure 5.7.

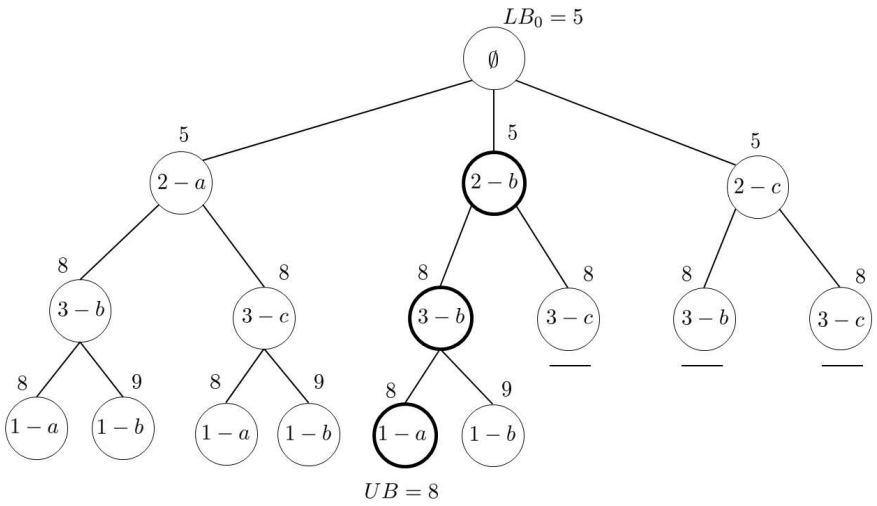


Figure 5.7. Complete branch and bound tree of the Slave Algorithm

The first rational assignment is $(2-b)$, $(3-b)$ and $(1-a)$ which corresponds to solution $z_a = 1$, $z_b = 7$ and $z_c = 0$ of cost 24. All the other nodes with $LB_{ij} \geq 8$ are pruned and the algorithm terminates with a bilevel-feasible assignment.

The solution of $BFLP$ is: $x_a = x_b = x_c = 1$, $z_a = 1\, z_b = 7\, z_c = 0$ and $y_{1a} = y_{3b} = 0.5\, y_{2b} = 1$. The cut introduced at the end of this iteration is $x_1 + x_2 + x_3 \leq 2$. Note that facility c is open but is not used, thus another bilevel-feasible solution is obtained by setting $x_c = 0$ and another cut can be added to $BFLP$, namely $x_1 + x_2 - x_3 \leq 1$.

5.2.5 Computational results

In this section we describe the computational results of the proposed branch and cut algorithm. These results are obtained by solving a set of benchmark test problems existing in the literature.

Our implementation environment consists of a PC Pentium Core 2 Duo with a 2GHz processor and 1GB RAM. The C programming is used to code both the branch and cut algorithm and the subroutine Slave Algorithm, the AMPL language and the solver CPLEX 12.3 are used to solve SLR. The benchmark instances used for the computational analysis are:

- 24 facilities and 50 clients, proposed by Kuehn and Hamburger [91], called KH_24
- 55 facilities and 55 clients, proposed by Swain [141], called S_55
- 49 facilities and 49 clients, proposed by Daskin [53], called D_49
- 88 facilities and 88 clients, proposed by Daskin [53], called D_88
- 150 facilities and 150 clients, proposed by Daskin [53], called D_150.

Test problems available in the literature were not completely suitable for our model and we added the price p_i paid by each client for the service and the capacity cost g that we assumed identical for all the facilities. For each test problem we created two different scenario: S' in which $p_i = \frac{1}{n\,m} \sum_{i \in I} \sum j \in J c_{ij}$, and S'' in which $p_i = \max_{i \in I,\, j \in J} c_{ij}$. In S' the price is set equal to the average value of costs c_{ij}, while in S'' the price paid by each client is greater than or equal to the assignment costs, i.e. each assignment $(i-j)$ yields a non negative

profit. We changed the value of parameter λ in the range $\{1,\ 0.75,\ 0.5,\ 0.25\}$: the higher is the value of λ, the tighter are the social equality constraints, i.e. the leader's requirements. In other words, by setting λ it is possible to simulate different levels of interaction between the decision makers: for $\lambda = 1$ the leader requires that all client's demand is satisfied, while for $\lambda = 0$ the leader can not influence the follower's behaviour at all, thus the bilevel model is meaningless and this case is not taken into account.

Finally, for the stopping criteria we fixed a maximum number of iterations $max\_iter = 500$ and an optimality gap of 25%.

In Tables 5.14-5.17 all the results are listed. Each benchmark problem is solved in both scenario S' and S'' and the tables list the value of parameter λ, the value $lead$ of the solution computed by the branch and cut algorithm in K€, the number of open facilities, the number of iterations, a lower bound lb in K€, the CPU time in seconds and the optimality gap between $lead$ and lb.

scenario	*instance*	λ	$lead(K€)$	*open*	*iter*	$lb(K€)$	CPU	*gap*
S'	KH_24	1	889.20	2	1	889.20	1.48	0.00%
	KH_24	0.75	835.00	4	434	685.52	949.73	21.81%
	KH_24	0.5	765.98	4	500	467.01	1121.97	64.02%
	KH_24	0.25	860.87	4	500	248.51	1112.75	246.42%
S''	KH_24	1	881.52	1	1	881.52	1.38	0.00%
	KH_24	0.75	826.83	2	240	670.52	395.41	23.31%
	KH_24	0.5	764.65	2	500	459.51	956.22	66.40%
	KH_24	0.25	702.46	2	500	241.01	969.82	191.47%

Table 5.14. Results for Kuehn and Hamburger problem

scenario	*instance*	λ	$lead(K€)$	*open*	*iter*	$lb(K€)$	CPU	*gap*
S'	S_55	1	2317.50	3	1	2317.50	3.46	0.00%
	S_55	0.75	2226.25	4	500	1739.88	1832.47	27.95%
	S_55	0.5	2167.75	4	500	1161.25	1862.46	86.67%
	S_55	0.25	2074.38	4	500	582.63	1824.03	256.04%
S''	S_55	1	2315.50	1	1	2315.50	2.53	0.00%
	S_55	0.75	2285.75	2	500	1737.88	1411.05	31.53%
	S_55	0.5	2255.00	2	500	1159.25	1377.55	94.52%
	S_55	0.25	2224.25	2	500	580.63	1312.84	283.08 %

Table 5.15. Results for Swain problem

The first remarkable result is that comparing scenario S' and S'' the number of open facilities is always larger in the first scenario. According to the definition of p, this shows that the model is sensitive to the price value and changes as expected: for small value of p the follower does not consider profitable to serve clients very far from the open facilities (i.e. a high assignment cost), thus the leader is obliged to open more facilities for satisfying the equality requirements. A similar result does not occur for the CPU time: indeed, for problems K_24 and

$scenario$	$instance$	λ	$lead(K€)$	$open$	$iter$	$lb(K€)$	CPU	gap
S'	D_49	1	752.92	2	1	752.92	1.65	0.00%
	D_49	0.75	729.59	3	500	575.24	1635.54	26.83%
	D_49	0.5	701.80	3	500	389.94	1669.58	79.98%
	D_49	0.25	674.01	3	500	204.63	1703.43	229.37%
S''	D_49	1	745.05	1	1	745.05	1.67	0.00%
	D_49	0.75	731.15	1	500	569.75	1908.85	28.33%
	D_49	0.5	717.24	1	500	384.45	1906.96	86.57%
	D_49	0.25	703.34	1	500	199.12	1916.7	253.22%

Table 5.16. Results for Daskin problem with 49 facilities

$scenario$	$instance$	λ	$lead(K€)$	$open$	$iter$	$lb(K€)$	CPU	gap
S'	D_88	1	135.67	2	1	135.67	7.02	0.00%
	D_88	0.75	134.09	3	500	102.66	4582.82	30.61%
	D_88	0.5	131.91	3	500	69.02	4573.44	91.13%
	D_88	0.25	129.72	3	500	35.37	4593.34	266.79%
S''	D_88	1	134.85	1	1	134.85	5.90	0.00%
	D_88	0.75	133.67	1	500	68.41	6164.27	95.40%
	D_88	0.5	132.21	1	500	68.41	6266.23	93.25%
	D_88	0.25	130.74	1	500	34.76	5913.91	276.11%

Table 5.17. Results for Daskin problem with 88 facilities

$scenario$	$instance$	λ	$lead(K€)$	$open$	$iter$	$lb(K€)$	CPU	gap
S'	D_150	1	176.72	2	1	176.72	303.09	0.00%
	D_150	0.75	174.70	2	500	133.04	10410.55	31.31%
	D_150	0.5	172.67	2	500	89.36	10139.75	93.23%
	D_150	0.25	170.65	2	500	45.68	9114.42	273.57%
S''	D_150	1	175.72	1	1	175.72	400.14	0.00%
	D_150	0.75	174.80	1	500	132.04	16007.43	32.39%
	D_150	0.5	173.16	1	500	88.36	18193.70	95.97%
	D_150	0.25	171.87	1	500	44.68	16826.92	284.67%

Table 5.18. Results for Daskin problem with 150 facilities

S_55 the computational time required in S' is larger than in S'', while for Daskin problems we can note the opposite result. Notwithstanding, the optimality gap obtained in S' is better than in S'' for all instances but one (K_24 with $\lambda = 0.25$).

Looking at parameter λ, it is important to note that it plays a crucial role in the model, as above mentioned. For $\lambda = 1$ the problem $BFLP$ is solved in one iteration and the optimal solution is found in a CPU time that is

considerably smaller than the other cases: two orders of magnitude smaller for K_24 and S_55, three orders for D_49 and D_88 and one order for D_150. With $\lambda = 1$ the follower's feasible set is significantly reduced as he is obliged to satisfy all the demands. For this reason, SLR provides the optimal solution of the problem.

Reducing the value of λ the quality of the solution dramatically worsen while the general performance of the algorithm is almost steady. Although the CPU time is not remarkably different when λ changes from 0.75 to 0.25, the optimality gap always increases, except D_88 in scenario S''. Looking at the value of the solution and at the number of open facilities we can note a similar behaviour of the CPU time. The main reason is that the lower bound computed solving SLR is not sufficiently tight especially for small values of λ. Computing a lower bound of poor quality does not allow to halt the algorithm before the maximum number of iterations is reached: indeed, in all but two instances, max_iter iterations are performed when λ is not equal to 1. This suggests that the optimality gap may be high due to a weak lower bound rather than a bilevel-feasible solution far from the optimum. Recall that the lower bound of $BFLP$ is computed by solving SLR in which the follower's objective function is dropped. Unfortunately, this procedure does not provide a good relaxation on benchmark instances used and its quality strictly depends on the value of parameter λ. Future works may be directed to define new valid relaxations of $BFLP$ in order to compute a tighter lower bound.

5.2.6 Conclusions

In this section we addressed a CFLP that was formulated as a discrete–continuous bilevel linear programming problem. The leader controls the investments for the facilities, i.e. he decides which facilities open and their capacities, while the follower is in charged of the assignment of clients to facilities. The main assumption of the model is that the leader has to make his decision now and, once the facilities are open, he can not control the follower nor oblige him to serve all clients. For this reason, the leader pursues both an economical objective, i.e. the minimization of the investment cost, and a social equality objective, i.e. he wants every client's demand to be satisfied beyond a certain threshold λ. On the other hand, the follower assigns clients to facilities only with the objective of maximizing his profit. Given a set of open facilities and capacities, the leader can foresee the optimal solution of the follower. Thus if the latter does not satisfy the equality requirements, the leader must increase the capacities or the number of facilities.

This model shows a clear hierarchical structure in which the two decision makers affect each other and bilevel programming is particularly suitable for this kind of problems. To our knowledge in the literature there are not similar applications of bilevel programming to CFLP. Our model is a DCBLP in which there are mixed-integer variables under control of the leader and continuous variables under control of the follower.

We proposed a branch and cut based algorithm that works according to the following rationale. A lower bound is computed solving the single level relaxation of the problem and it is checked its bilevel-feasibility. If the solution is not bilevel-feasible, it is possible to increase the capacity of the open facilities and compute a new bilevel-feasible solution. For computing the minimum amount of capacity that has to be increased to restore bilevel-feasibility, we solve a slave bilevel problem by means of a subroutine. This subroutine, that is embedded in the algorithm, is a branch and bound approach that considers all the possible assignments of clients to pen facilities and always provides a bilevel-feasible solution. After that, a cut is introduced to discard this solution and the algorithm iterates from scratch. The algorithm is able to find the optimal solution in theory, notwithstanding we introduced a maximum number of iterations and an optimality gap in order to cope with large size instances.

The algorithm was tested on a set of benchmark problems present in the literature. Computational results show that parameter λ plays a key role for the performance of the algorithm. From a mathematical standpoint λ

affects the relation between the two decision makers and represents the authority of the leader: the higher is λ, the tighter are the leader's requirements. For $\lambda = 1$ the single level relaxation provides the optimal solution, while for decreasing values the optimality gap and the CPU time dramatically increase. This behaviour stems from the poor quality of the lower bound provided by the single level relaxation. For this reason the algorithm reaches the maximum number of iterations in almost all the instances solved and the quality of the solution found is not significant. Future research may be addressed to compute tighter lower bounds and deeper valid inequalities for this bilevel facility location problem in order to speed up the resolution and have a more significant assessment of the solution's quality provided by the algorithm.

Chapter 6

Conclusion

In this dissertation we focused our attention on bilevel programming and in particular on the subclass of Bilevel Linear Problems (BLPs) with a subset or all the variables required to be integer. Bilevel programming represents a relatively modern area of research: although the first example of bilevel linear problems are not vey recent, in the last two decades these probems have received an increasing interest from the scientific community for several reasons. Bilevel problems, even in the simplest linear version, are very challenging problems with particular mathematical properties. Despite their inner computational complexity, they represent very powerful mathematical programming approaches and allow to model real life problems in a novel and more realistic way compared to traditional single level models.

The main characteristic of bilevel problems is the presence of two decision makers hierarchically related within the same mathematical program. The two decision makers, denoted as leader and follower, optimize their own objective functions affecting each other. Once the leader makes a choice, the follower reacts according to his objective function. Hence the leader has to take the follower's best reaction into account in order to minimize or maximize his objectives function. From a mathematical point of view, a bilevel problem is comprised of two nestled problems in which the optimal solution of the inner problem (follower problem) is a feasible solution for the outer problem (leader problem). BLPs are closely related to other well known problems, such as MPECs, Stackelberg games, Stochastic Programming Problems or Multiobjective Programming Problems. For this reason, there are many different directions of research and investigation and this is one of the most interesting and promising feature of this class of problems. In this dissertation we focused on integer BLPs and we studied them trying to exploit their geometrical properties and their similarity to classical linear and integer linear problems. The most relevant part of this work investigates the impact of integer variables on BLPs: for the sake of clearness in Chapter 3 we studied BLPs with integer variables only in the upper level, while in Chapter 4 we extended the study to pure integer BLPs with both upper and lower level discrete problems. We believe that, despite their computational complexity, integer bilevel linear problems have not been sufficiently investigated despite their potentiality, especially for modelling integer and combinatorial problems with an inner hierarchical structure and multiple decision makers involved.

Basically, the computational complexity of bilevel problems stems from their nestled structure. Although the two problems are linear programming problems, the resulting BLP is a NP-hard problem and requires some additional notations. In Chapter 1 we introduced some basic definition such as follower's feasible set, rational set, inducible region and bilevel-feasible solution. Moreover we described the computational complexity of

bilevel problems, their relation to other similar problems and made an overview of all possible applications that have been proposed in the literature.

In Chapter 2 we examined in detail all the main mathematical and polyhedral properties of BLPs, we provided some additional definitions such as single level relaxation and single level reformulation. Great attention was dedicated to the impact of upper level constraints: in our opinion the role of this set of constraints, which determines the difference between reaction set and inducible region, is often overlooked in the literature. In the rest of the chapter we reported an overview of the state of the art that is rich of solution methods since BLPs are the simplest and most studied bilevel programming problems. Finally, we investigated the role of integer variables in bilevel formulation and we defined two main problems that represent the principal topic of this dissertation: Discrete–Continuous Bilevel Linear Problems (DCBLPs) and Discrete Bilevel Linear Problems (DBLPs).

In Chapter 3 we studied DCBLPs and proposed two different solution approaches. In the first part we started from an existing reformulation technique that relaxes the integer upper level variables by means of a penalized function. We proposed an improvement of the existing method, we tested and certificated that our approach is better than the existing one despite it still presents some drawbacks that do not allow to use it as an exact method and need more investigation. In the second part we proposed a new valid inequality that is based on some geometrical properties of BLPs. This inequality represents the effort to solve discrete BLPs starting from well known properties of the corresponding continuous BLPs as it happens between linear programs and integer linear programs. The proposed inequality is valid for BLPs and DCBLPs and was efficiently used to speed up the resolution of a generic DCBLP with encouraging results on small and medium size problems.

In Chapter 4 we analyzed DBLPs that are one of the most challenging class of BLPs. We followed two lines of research providing two different results: two new exact methods and two new heuristic approaches. In the first part of the chapter we proposed two new exact methods, a branch and cut and a cutting plane approach. Both the methods were tested and compared to a benchmark existing algorithm. Their computational performances, especially for the cutting plane, clearly show an improvement and a remarkable progress in the state of the art of solution methods for DBLPs. The second part of the chapter is dedicated to the description of two heuristic approaches based of the valid inequality proposed in Chapter 3. We show how it is possible to extend and readapt the inequality, that is not valid in general for DBLPs, and how to embed it within an existing method to design two new heuristics.

In Chapter 5 we described two applications of integer BLPs to real problems in the field of operation management. The first application is a Grid scheduling problem that is modelled as a DBLP and is solved by means of a tabu search heuristic that, on one side, exploits the bilevel structure of the problem and, on the other side, copes with the big size of real life instances. The second application is a capacitated facility location problem modelled as a DCBLP. We propose a branch and cut framework that is able to compute the optimal solution, in theory, but it is halted by suitable stopping criteria for solving real life instances within a reasonable computing time for the application. Both the applications represent innovative use of bilevel programming for such problems.

The study conducted on integer bilevel linear problems highlighted that this class of problems has not been sufficiently explored. We proposed new contributions in terms of exact methods and heuristics: some of them can be considered preliminary encouraging results, some other certainly represent promising directions

for future research. In our opinion it is still necessary to better investigate the connection between BLPs and other well known problems, especially Multiobjective Programming Problems. Some theoretical results about the equivalence of the latter two problems have been shown, but only in theory and not for the integer case. Bilevel programming may be potentially used to model and solve a wide range of interesting applications, although the lack of efficient solution methods limits their use. The development of new fast heuristics, well performing on big size instances, is a necessary direction for future research. Finally, it may be possible to exploit the theoretical results and contributions provided in this dissertation to design new valid inequalities, new reformulation approaches and new branch and bound schemes in order to readapt, as much as possible, well known methods of integer linear programming for solving integer bilevel linear problems.

Appendix A

Acronyms

BC	–	Branch and Cut algorithm
BFLP	–	Bilevel Facility Location Problem
BLP	–	Bilevel Linear Problem
CDBLP	–	Continuous–Discrete Bilevel Linear Problem
CFLP	–	Capacitated Facility Location Problem
CP	–	Cutting Plane algorithm
DBLP	–	Discrete Bilevel Linear Problem
DCBLP	–	Discrete–Continuous Bilevel Linear Problem
DR	–	DeNegre and Ralphs algorithm
ES	–	External Scheduler
FP(s)	–	Follower Problem
GCD	–	Greatest Common Divisor
GWA	–	Grid Workload Archive
HBC	–	Hybrid Branch and Cut algorithm
IR	–	Inducible Region
KKT	–	Karush–Kuhn–Tucker conditions
LB	–	Lower Bound
LMM	–	Liner Max-Min Problem
LP	–	Leader Problem
LS	–	Local Scheduler
MCP	–	Modified Cutting Plane algorithm
MIP	–	Mixed–Integer Programming
MPEC	–	Mathematical Programming with Equilibrium Constraints
RSLF	–	Relaxed Single Level Reformulation
SLF	–	Single Level Reformulation
SLR	–	Single Level Relaxation
SVP(x_j)	–	Slave Problem
TUM	–	Totally Unimodular
UB	–	Upper Bound
UFLP	–	Uncapacitated Facility Location Problem

Bibliography

[1] Ahuja, R. K., Magnanti, T. L., Orlin, J. B. (1993). *Network flows: theory, algorithms, and applications*. Prentice Hall.

[2] Anandalingam, G. (1988). A mathematical programming model of decentralized multi-level systems. *The Journal of the Operational Research Society*, 39(11), 1021–1033.

[3] Anandalingam, G., Mathieu, R., Pittard, C. L., Sinha, N. (1989). Artificial intelligence based approaches for solving hierarchical optimization problems. In Sharda, R., Golden, B. L., Wasil, E., Balci, O., Stewart, W. (Eds.) *Impacts of recent computer advances in operations research* , North–Holland, 289–301.

[4] Anandalingam, G., White, D. J. (1990). A solution method for the linear static Stackelberg problem using penalty functions. *IEEE Transactions on Automatic Control*, 35(10), 1170–1173.

[5] Akinc, U., Khumawala, B. M. (1977). An efficient branch and bound algorithm for the capacitated warehouse location problem. *Management Science*, 23(6), 585–594.

[6] Aksen, D., Akca, S. S., Aras, N. (2014). A bilevel partial interdiction problem with capacitated facilities and demand outsourcing. *Computers and Operations Research*, 41, 346–358.

[7] Aksen, D., Aras, N. (2012). A bilevel fixed charge location model for facilities under imminent attack. *Computers and Operations Research*, 39(7), 1364–1381.

[8] Aksen, D., Aras, N. (2013). A matheuristic for leader-follower games involving facility location-protection-interdiction decisions. In Talbi E. (Ed.) *Metaheuristics for bi-level optimization studies in computational intelligence*, Studies in Computational Intelligence, volume 482, Springer, 115–151.

[9] Alarie, S., Audet, C., Jaumard, B., Savard, G. (2001). Concavity cuts for disjoint bilinear programming. *Mathematical Programming*, 90(2), 373–398.

[10] Arroyo, J. M., Galiana, F. D. (2005). On the solution of the bilevel programming formulation of the terrorist threat problem. *IEEE Transactions on Power Systems*, 20(2), 789–797.

[11] Audet, C., Haddad, J., Savard, G. (2006). A note on the definition of a linear bilevel programming solution. *Applied Mathematics and Computation*, 181(1), 351–355.

[12] Audet, C., Haddad, J., Savard, G. (2007). Disjunctive cuts for continuous linear bilevel programming. *Optimization Letters*, 1(3), 259–267.

[13] Audet, C., Hansen, P., Jaumard, B., Savard, G. (1997). Links between linear bilevel and mixed 0-1 programming problems. *Journal of Optimization Theory and Applications*, 93(2), 273–300.

[14] Audet, C., Hansen, P., Jaumard, B., Savard, G. (1999). A symmetrical linear maxmin approach to disjoint bilinear programming. *Mathematical Programming*, 85(3), 573–592.

[15] Audet, C., Savard, G., Zghal, W. (2007). New branch-and-cut algorithm for bilevel linear programming. *Journal of Optimization Theory and Applications*, 134(2), 353–370.

[16] Averbakh, I. (2000). Minmax regret solutions for minimax optimization problems with uncertainty. *Operations Research Letters*, 27(2), 57–65.

[17] Balinski, M. L., Wolfe, P. (1963). On Benders decomposition and a plant location problem. Working paper-ARO 27. *Mathematica*, Princeton.

[18] Bard, J. F. (1983). An efficient point algorithm for a linear two-stage optimization problem. *Operations Research*, 31(4), 670–684.

[19] Bard, J. F. (1983). Coordination of a multidivisional firm through two levels of management. *Omega*, 11(5), 457–465.

[20] Bard, J. F. (1984). Optimality conditions for the bilevel programming problem. *Naval Research Logistics Quarterly*, 31(1), 13–26.

[21] Bard, J. F. (1998). *Practical bilevel optimization: algorithms and applications*. Kluwer Academic Publishers.

[22] Bard, J. F. (2001). Bilevel programming in management . *Encyclopedia of Optimization*, Kluwer Academic Publishers, volume 1, 173–177.

[23] Bard, J. F., Falk, J. E. (1982). An explicit solution to the multi-level programming problem. *Computers and Operations Research*, 9(1), 77–100.

[24] Bard, J. F., Moore, J. T. (1990). A branch and bound algorithm for the bilevel programming problem. *SIAM Journal on Scientific and Statistical Computing*, 11(2), 281–292.

[25] Bard, J. F., Plummer, J., Sourie, J. C. (2000). A bilevel programming approach to determining tax credits for biofuel production. *European Journal of Operational Research*, 120(1), 30–46.

[26] Ben-Ayed, O. (1993). Bilevel linear programming. *Computers and Operations Research*, 20(5), 485–501.

[27] Ben-Ayed, O., Blair, C. (1990). Computational difficulties of bilevel linear programming. *Operations Research*, 38(3), 556–560.

[28] Ben-Ayed, O., Blair, C., Boyce, D., LeBlanc, L. J. (1992). Construction of a real-world bilevel linear programming model of the highway network design problem. *Annals of Operations Research*, 34(1), 219–254.

[29] Ben-Ayed, O., Boyce, D., Blair, C. (1988). A general bilevel linear programming formulation of the network design problem. *Transportation Research Part B*, 22(4), 311–318.

[30] Bialas, W. F. (1989). Cooperative $n-$person Stackelberg games. In *Proceedings of the* 28^{th} *IEEE Conference on Decision and Control*, 3, 2439–2444.

[31] Bialas, W. F., Chew, M. N. (1982). Coalition formation in $n-$person Stackelberg games. In *Proceedings of the* 21^{st} *IEEE Conference on Decision and Control*, 21, 669–672.

[32] Bialas, W. F., Karwan, M. H. (1978). Multilevel linear programming. Technical Report 78-1, State University of New York at Buffalo, Operations Research Program.

[33] Bialas, W. F., Karwan, M. H. (1982). On two-level optimization. *IEEE Transactions on Automatic Control*, 27(1), 211–214.

[34] Bialas, W. F., Karwan, M. H. (1984). Two-level linear programming. *Management Science*, 30(8), 1004–1020.

[35] Bianco, L., Caramia, M., Giordani, S. (2009). A bilevel flow model for hazmat transportation network design. *Transportation Research Part C*, 17, 175–196.

[36] Birge, J. R., Louveaux, F. (1997). *Introduction to stochastic programming* Springer Series in Operations Research.

[37] Bracken, J., McGill, J. T. (1973). Mathematical programs with optimization problems in the constraints. *Operations Research*, 21(1), 37–44.

[38] Bracken, J., McGill, J. T. (1978). Production and marketing decisions with multiple objectives in a competitive environment. *Journal of Optimization Theory and Applications* , 24(2), 449–458.

[39] Brotcorne, L., Marcotte, P., Savard, G. (2008). Bilevel programming: the Montreal school. *INFOR*, 48(4), 231–246.

[40] Buyya, R. (2002). Economic-based distributed resource management and scheduling for Grid computing. Ph.D. thesis, School of Computer Science and Software Engineering, Monash University, Melbourne.

[41] Buyya, R., Abramson, D., Giddy, J., Stockinger, H. (2002). Economic models for resource management and scheduling in Grid computing. *Concurrency and Computation: Practice and Experience*, 14(13-15), 1507–1542.

[42] Candler, W., Norton, R. (1977). Multilevel programming. Technical Report 20. World Bank Development Research Center, Washington D.C.

[43] Candler, W., Townsley, R. (1982). A linear two-level programming problem. *Computers and Operations Research*, 9(1), 59–76.

[44] Caramia, M., Giordani, S. (2011). An economic model for resource allocation in Grid computing. *Operations Research*, 59(4), 956–972.

[45] Casanova, H., Dongarra, J. (1997). NetSolve: A network server for solving computational science problems. *International Journal of Supercomputing Applications and High Performance Computing*, 11(3), 212–223.

[46] Cassidy, R. G., Kirby, M. J. L., Raike, W. M. (1971). Efficient distribution of resources through three levels of government. *Management Science*, 17(8), 462–473.

[47] Chapin, S. J., Katramatos, D., Karpovich, J., Grimshaw, A. S. (1999). The legion resource management system. *Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science*, 1659, 162–178.

[48] Charnes, A., Clower, W., Kortanek, K. O. (1967). Effective control through coherent decentralization with preemptive goals. *Econometrica* 35(2), 249–319.

[49] Colson, B., Marcotte, P., Savard, G. (2007). An overview of bilevel optimization. *Annals of Operations Research*, 153(1), 235–256.

[50] Cornuejols, G., Fisher, M., Nemhauser, G. L. (1977). Location of bank accounts to optimize float: an analytic study of exact and approximate algorithms. *Management Science*, 23(8), 789–810.

[51] Cornuejols, G. , Nemhauser, L. G., Wolsey, L. A. (1990). The uncapacitated facility location problem. In Mirchandani, P., Francis, R. (Eds.) *Discrete location theory*, John Wiley and Sons Inc., 119–171.

[52] Danskin, J. W. (1966). The theory of max-min with applications. *SIAM Journal of Applied Mathematics*, 14(4), 641–664.

[53] Daskin, M. S. (1995). *Network and discrete location: models, algorithms, and applications*. John Wiley and Sons Inc.

[54] Dempe, S. (2002). *Foundation of bilevel programming*. Kluwer Academic Publishers.

[55] Dempe, S., Dutta, J. (2010). Is bilevel programming a special case of a mathematical program with complementarity constraints? *Mathematical Programming*, 131(1-2), 37–48.

[56] DeNegre, S. T. (2011). Interdiction and discrete bilevel linear programming. Ph.D. thesis, Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, Pennsylvania.

[57] DeNegre, S. T., Ralphs, T. K. (2009). A branch-and-cut algorithm for integer bilevel linear programs. *Operations Research and Cyber-Infrastructure*, 47(2), 65–78.

[58] Efroymson, M. A., Ray, T. L. (1966). A branch-bound algorithm for plant location. *Operations Research*, 14(3), 361–368.

[59] Erlenkotter, D. (1978). A dual-based procedure for uncapacitated facility location. *Operations Research*, 26(6), 992–1009.

[60] Falk, J. E. (1973). A linear max-min problem. *Mathematical Programming* 5(1), 169–188.

[61] Fliege, J., Vicente, N. (2006). Multicriteria approach to bilevel optimization. *Journal of Optimization Theory and Applications*, 131(2), 209–225.

[62] Fortuny-Amat, J., McCarl, B. (1981). A representation and economic interpretation of a two-level programming problem. *Journal of the Operational Research Society*, 32(9), 783–792.

[63] Foster, I., Kesselmanm C. (2004). *The Grid 2: blueprint for a new computing infrastructure*. Morgan Kaufmann Publishers.

[64] Fülöp, J. (1993). On the equivalence between a linear bilevel programming problem and linear optimization over the efficient set. Technical Report WP 93-1, Laboratory of Operations Research and Decision Systems, Computer and Automation Institute, Hungarian Academy of Sciences.

[65] Gallo, G., Ülkücü, A. (1977). Bilinear programming: an exact algorithm. *Mathematical Programming*, 12(1), 173–194.

[66] Gao, Z., Wu, J., Sun, H. (2005). Solution algorithm for the bi-level discrete network design problem. *Transportation Research Part B*, 39(6), 479–495.

[67] Garey, M. R., Johnson, D. S. (1979). *Computers and intractability: a guide to the theory of NP-completeness*. W. H. Freeman.

[68] Gendreau, M., Marcotte, P., Savard, G. (1996). A hybrid tabu-ascent algorithm for the linear bilevel programming problem. *Journal of Global Optimization*, 8(3), 217–233.

[69] Geoffrion, A. M. (1968). Proper efficiency and the theory of vector maximization. *Journal of Mathematical Analysis and Applications*, 22(3), 618–630.

[70] Geoffrion, A. M., Graves, G. W. (1974). Multicommodity distribution system design by Benders decomposition. *Management Science*, 20(5), 822–844.

[71] Gzara, F. (2013). A cutting plane approach for bilevel hazardous material transport network design. *Operations Research Letters*, 41(1), 40–46.

[72] Hakimi, S. L. (1964). Optimum locations of switching centers and the absolute centers and medians of a graph. *Operations Research*, 12(3), 450–459.

[73] Hakimi, S. L. (1965). Optimum distribution of switching centers in a communication network and some related graph theoretic problems. *Operations Research*, 13(3), 462–475.

[74] Hansen, P., Jaumard, B., Savard, G. (1992). New branch-and-bound rules for linear bilevel programming. *SIAM Journal on Scientific and Statistical Computing*, 13(5), 1194–1217.

[75] Haurie, A., Savard, G., White, D. (1990). A note on: an efficient point algorithm for a linear two-stage optimization problem. *Operations Research*, 38(3), 553–555.

[76] Hobbs, B. F., Nelson, S. K. (1992). A nonlinear bilevel model for analysis of electric utility demand-side planning issues. *Annals of Operations Research*, 34(1), 255–274.

[77] Iosup, A., Li, H., Jan, M., Anoep, S., Dumitrescu, C., Wolters, L., Epema, D. H. J. (2008). The Grid Workloads Archive. *Future Generation Computer Systems*, 24(7), 672–686.

[78] Jeroslow, R. G. (1985). The polynomial hierarchy and a simple model for competitive analysis. *Mathematical Programming*, 32(2), 146–164.

[79] Jùdice, J. J., Faustino, A. M. (1992). A sequential LCP method for bilevel linear programming. *Annals of Operations Research*, 34(1), 89–106.

[80] Kalantari, B., Rosen, J. B. (1982). Penalty for zero-one integer equivalent problem. *Mathematical Programming*, 24(1), 229–232.

[81] Kall, P., Wallace, S. W. (1994). *Stochastic programming*, Wiley-Interscience.

[82] Kapadia, N. H., Fortes, J. A. B. (1999). PUNCH: an architecture for web-enabled wide-area network-computing. *Cluster Computing*, 2(2), 153–164.

[83] Kara, B. Y., Verter, V. (2004). Designing a road network for hazardous materials transportation. *Transportation Science*, 38(2), 188–196.

[84] Karlof, J. K., Wang, W. (1996). Bilevel programming applied to the flowshop scheduling problem. *Computers and Operations Research*, 23(5), 443–451.

[85] Karp, R. M. (1972). Reducibility among combinatorial problems. In Miller, R. E., Tatcher, J. W. (Eds.) *Complexity of computer computations*, Plenum Press, 85–103.

[86] Khumawala, B. M. (1972). An efficient branch-and-bound algorithm for the warehouse location problem. *Maagement Science*, 18(12), 718–731.

[87] Kis, T., Kovács, A. (2009). On bilevel machine scheduling problems. In *Proceedings MAPSP 2009, Workshop on Models and Algorithms for Planning and Scheduling Problems*, 116–118.

[88] Kochetov, Y., Kochetova, N., Plyasunov, A. (2013). A matheuristic for the leader-follower facility location and design problem. In *Proceedings of the 10^{th} Metaheuristics International Conference (MIC 2013)*, 1–3.

[89] Konno, H. (1971). Bilinear programming: Part I. Algorithm for solving bilinear programs. Technical Report 71-9, Stanford University.

[90] Krarup, J., Pruzan, P. M. (1983). The simple plant location problem: survey and synthesis. *European Journal of Operational Research*, 12(1), 36–81.

[91] Kuehn, A. A., Hamburger, M. J. (1963). A heuristic program for location warehouses. *Management Science*, 9(4), 643–666.

[92] Kurowski, K., Nabrzyski, J., Oleksiak A., Weglarz, J. (2008). Multicriteria approach to two-level hierarchy scheduling in Grids. *Journal of Scheduling*, 11(5), 371–379.

[93] Kurowski, K., Oleksiak A., Weglarz, J. (2010). Multicriteria, multi-user scheduling in Grids with advance reservation. *Journal of Scheduling*, 13(5), 493–508.

[94] Labbé, M., Marcotte, P., Savard, G. (1998). A bilevel model of taxation and its application to optimal highway pricing. *Management Science*, 44(12), 1595–1607.

[95] Liberatore, F., Scaparra, M. P., Daskin, M. S. (2011). Analysis of facility protection strategies against an uncertain number of attacks: the stochastic R-interdiction median problem with fortification. *Computers and Operations Research*, 38(1), 357–366.

[96] Lignola, M. B., Morgan, J. (1995). Topological existence and stability for Stackelberg problems. *Journal of Optimization Theory and Applications*, 84(1), 145–169.

[97] Litzkow, M., Livny, M., Mutka, M. W. (1988). Condor - a hunter of idle workstations. In *Proceedings of the 8^{th} International Conference of Distributed Computing Systems (ICDCS 1988)*, 104–111.

[98] Lodi, A., Ralphs, T. K. (2009). Bilevel programming and maximally violated valid inequalities. In *Proceedings of the Cologne Twente Workshop on Graphs and Combinatorial Optimization*, 125–134.

[99] Loridan, P., Morgan, J. (1996). Weak via strong Stackelberg problem: new results. *Journal of Global Optimization*, 8(3), 263–287.

[100] Losada, C., Scaparra, M. P., Church, R. L. (2010). On a bi-level formulation to protect uncapacitated p-median systems with facility recovery time and frequent disruptions. *Electronic Notes in Discrete Mathematics*, 36, 591–598.

[101] Louveaux, F. V., Peeters, D. (1992). A dual-based procedure for stochastic facility location. *Operations Research*, 40(3), 564–573.

[102] Lukač, Z., Šorić, K., Rosenzweig, V. V. (2008). Production planning problem with sequence dependent setups as a bilevel programming problem. *European Journal of Operational Research*, 187(3), 1504–1512.

[103] Luo, Z. Q., Pang, J. S., Ralph, D. (1996). *Mathematical programs with equilibrium constraints*. Cambridge University Press.

[104] Mahajan, A. (2010). Presolving mixed-integer linear programs. *Preprint ANL/MCS-P1752-0510, Mathematics and Computer Science Division*.

[105] Mansi, R., Alves, C., de Carvalho, J. M. V., Hanafi, S. (2012). An exact algorithm for bilevel 0-1 knapsack problems. *Mathematical Problems in Engineering*, Hindawi Publishing Corporation, volume 2012.

[106] Marcotte, P. (1986). Network design problem with congestion effects: a case of bilevel programming. *Mathematical Programming*, 34(2), 142–162.

[107] Marcotte, P., Savard, G. (1991). A note on the Pareto optimality of solutions to the linear bilevel programming problem. *Computers and Operations Research*, 18(4), 355–359.

[108] Marcotte, P., Savard, G. (2001). Bilevel programming: applications. *Encyclopedia of Optimization*, Kluwer Academic Publishers, volume 1, 158–159.

[109] Marcotte, P., Savard, G. (2005). Bilevel programming: a combinatorial perspective. In Avis, D., Hertz, A., Marcotte, O. (Eds.) *Graph theory and combinatorial optimization*, Kluwer Academic Publisher.

[110] Mathieu, R., Pittard, L., Anandalingam, G.(1994). Genetic algorithm based approach to bi-level linear programming. *RAIRO - Operations Research - Recherche Opérationnelle*, 28(1), 1–21.

[111] Mattia, S. (2012). Separating tight metric inequalities by bilevel programming. *Operations Research Letters*, 40(6), 568–572.

[112] Mersha, A. G., Dempe, S. (2005). Linear bilevel programming with upper level constraints depending on the lower level solution. *Applied Mathematics and Computation*, 180(1), 247–254.

[113] Migdalas, A. (1995). Bilevel programming in traffic planning: models, methods and challenge. *Journal of Global Optimization*, 7(4), 381–405.

[114] Moore, J. T., Bard, J. F. (1990). The mixed integer linear bilevel programming problem. *Operations Research*, 38(5), 911–921.

[115] Nabrzyski, J., Schopf, J. M., Weglarz, J. (2004). *Grid resource management: state of the art and future trends*. Kluwer Academic Publisher.

[116] Nash, J. (1951). Non-cooperative games. *Annals of Mathematics*, 54(2), 286–295.

[117] Nauss, R. M. (1978). An improved algorithm for the capacitated facility location problem. *Journal of Operational Research Society*, 29(12), 1195–1201.

[118] Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V. (2007). *Algorithmic game theory*, Cambridge University Press.

[119] Osborne, M. J. (2003). *An introduction to game theory*. Oxford University Press.

[120] Özaltin, O. Y., Prokopyev, O. A., Schaefer, A. J. (2010). The bilevel knapsack problem with stochastic right-hand sides. *Operations Research Letters*, 38(4), 328–333.

[121] Papavassilopoulos, G. (1982). Algorithms for static Stackelberg games with linear costs and polyhedral constraints. In *Proceedings of the* 21^{st} *IEEE Conference on Decision and Control*, 2, 647–652.

[122] Pareto, V. (1986). *Cours d'economie politique*. Rouge, Lausanne.

[123] Parraga, F. A. (1981). Hierarchical programming and applications to economic policy. Ph.D. thesis, Department of Systems and Industrial Engineering, University of Arizona, Tucson, Arizona.

[124] Ranganathan, K., Foster, I. (2002). Decoupling computation and data scheduling in distributed data-intensive applications. In *Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing (HPDC-11)*, 352–358.

[125] Roy, B. (1968). Classement et choix en présence de point de vue multiples (La méthod ELECTRE). *Revue Française d'Automatique, d'Informatique et de Recherche Opérationnelle* , 2(1), 57–75.

[126] Sahin, K., Gümüs, Z., Ciric, A. (2001). Bilevel programming: applications in engineering. *Encyclopedia of Optimization*, Kluwer Academic Publishers, volume 1, 160–163.

[127] Savard, G. (1989). Contribuition à la programmation mathématique à deux niveaux. Ph.D. thesis, École Polytechnique de Montréal.

[128] Scaparra, M. P., Church, R. L. (2008). A bilevel mixed-integer program for critical infrastructure protection planning. *Computers and Operations Research*, 35(6), 1905–1923.

[129] Scaparra, M. P., Church, R. L. (2008). An exact solution approach for the interdiction median problem with fortification. *European Journal of Operational Research*, 189(1), 76–92.

[130] Selten, R. (1975). Reexamination of the perfectness concept for equilibrium points in extensive games. *International Journal of Game Theory*, 4(1), 25–55.

[131] Sherali, H. D., Soyster, A. L., Murphy, F. H. (1983). Stackelberg-Nash-Cournot equilibria: characterizations and computations. *Operations Research*, 31(2), 253–276.

[132] Shi, C., Lu, J., Zhang, G. (2005). An extend Kuhn-Tucker approach for linear bilevel programming. *Applied Mathematics and Computation*, 162(1), 51–63.

[133] Shi, C., Lu, J., Zhang, G. (2005). An extended Kth-best approach for linear bilevel programming. *Applied Mathematics and Computation*, 164(4), 843–855.

[134] Shi, C., Lu, J., Zhang, G., Zhou, H. (2006). An extended branch and bound algorithm for linear bilevel programming. *Applied Mathematics and Computation*, 180(2), 529–537.

[135] Shi, C., Zhang, G., Lu, J. (2005). On the definition of linear bilevel programming solution. *Applied Mathematics and Computation*, 160(1), 169–176.

[136] Shiquan, W. U., Yang, C., Marcotte, P. (1998). A cutting plane method for linear bilevel programs. *System Sciences and Mathematical Sciences*, 11(2), 125–133.

[137] Smith, J. C., Lim, C., Alptekinoğlu, A. (2009). New product introduction against a predator: a bilevel mixed-integer programming approach. *Naval Research Logistics*, 56(8), 714–729.

[138] Snyder, L. V. (2006). Facility location under uncertainty: a review. *IIE Transactions*, 38(7), 537–554.

[139] Stackelberg, H. (1952). *The theory of market economy*. Oxford University Press.

[140] Su, A., Berman, F., Wolski, R., Strouta, M. M. (1999). Using AppLeS to schedule simple SARA on the computational Grid. *International Journal of High Performance Computing Applications*, 13(3), 253–262.

[141] Swain, R. (1971). A decomposition algorithm for a class of facility location problems. Ph.D. thesis, Cornell University, Ithaca, New York.

[142] Tchernykh A., Ramarez J., Avetisyan A., Kuzjurin N., Grushin D., Zhuk S. (2006). Two level job-scheduling strategies for a computational grid. *Parallel Processing and Applied Mathematics, Lecture Notes in Computer Science*, 3911, 774–781.

[143] Ünlü, G. (1987). A linear bilevel programming algorithm based on bicriteria programming. *Computers and Operations Research*, 14(2), 173–179.

[144] Verter, V. (2011). Uncapacitated and capacitated facility location problems. In Eiselt, H. A. , Marianov, V. (Eds.), *Foundations of location analysis*, Springer, 25–37.

[145] Vicente, L. N., Calai, P. H. (1994). Bilevel and multilevel programming: a bibliography review. *Journal of Global Optimization*, 5(3), 291–306.

[146] Vicente, L. N., Savard, G., Judice, J. J. (1994). Descent approaches for quadratic bilevel programming. *Journal of Optimization Theory and Applications*, 81(2), 379–399.

[147] Vicente, L. N., Savard, G., Judice, J. J. (1996). Discrete linear bilevel programming problem. *Journal of Optimization Theory and Applications*, 89(3), 597–614.

[148] Wen, U. P., Hsu, S. T. (1989). A note on a linear bilevel programming algorithm based on bicriteria programming. *Computers and Operations Research*, 16(1),79–83.

[149] Wen, U. P., Hsu, S. T. (1991). Linear bi-level programming problems – A review. *Journal of the Operational Research Society*, 42(2), 125–133.

[150] Wen, U. P., Huang, A. D. (1996). A simple tabu search method to solve the mixed-integer linear bilevel programming problem. *European Journal of Operational Research*, 88(3), 563–571.

[151] Wen, U. P., Yang, Y. H. (1990). Algorithms for solving the mixed integer two-level linear programming problem. *Computers and Operations Research*, 17(2), 133–142.

[152] White, D. J., Anandalingam, G. (1993). A penalty function approach for solving bi-level linear programs. *Journal of Global Optimization*, 3(4), 397–419.