UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA INDUSTRIAL

BENDERS DECOMPOSITION BASED ALGORITHMS FOR GENERAL AND SECURITY STACKELBERG GAMES

TESIS PARA OPTAR AL GRADO DE
MAGÍSTER EN GESTIÓN DE OPERACIONES

INGRID ALEJANDRA ARRIAGADA FRITZ

PROFESOR GUÍA:
FERNANDO ORDÓÑEZ PIZARRO

MIEMBROS DE LA COMISIÓN:
MARTINE LABBÉ
ANDRÉS WEINTRAUB POHORILLE

Este trabajo ha sido parcialmente financiado por CONICYT a través del proyecto FONDECYT Nº 1171419.

SANTIAGO DE CHILE
2021

THESIS ABSTRACT FOR THE
DEGREE OF: MAGÍSTER EN GESTIÓN DE OPERACIONES
AUTHOR: INGRID ALEJANDRA ARRIAGADA FRITZ
YEAR: 2021
ADVISOR: FERNANDO ORDÓÑEZ PIZARRO

BENDERS DECOMPOSITION BASED ALGORITHMS FOR GENERAL AND SECURITY STACKELBERG GAMES

Stackelberg games model situations where there exist more than one decision maker in a leader-follower interaction, where the leader commits to a decision first and then the follower makes a decision. When the interactions are in a security context with the leader modeling the defender and the follower the attacker, it is referred to as Stackelberg security game. Stackelberg games are widely used for applications in economics, transportation, biology, finance, security, among many others. In real world, there exist security applications used to protect ports, borders, airports and other critical targets.

Due to the use of general and security Stackelberg games in real world applications, it becomes important to solve these problems efficiently. Thereby, users can make decisions when required, every day or every hour. This need for efficiency has produced different formulations and solution methods for these Stackelberg games. In this thesis we contribute to the work on mixed integer formulations of these Stackelbeg games and decomposition algorithms to solve them. Specifically we consider Benders cuts generated from the strong formulation, and normalized Benders cuts as introduced by [29]. For that reason, this thesis is focused in algorithms performance.

Recent research has studied the efficiency of different mixed integer formulations for Stackelberg games, including Stackelberg security games. A central finding is that the formulation with the smallest integrality gap, which considers more variables and constraints, is the one that has the linear relaxation that is most difficult to solve. This trade-off makes it unclear which formulation is preferred to solve specific instances and suggests strengthening the weak formulation (WF) with cuts from the strong formulation (SF).

This work studies the use of Benders cuts on the WF that are generated from the SF. In particular, we investigate the effect of different regularization methods, the use of normalized Benders cuts, and cutting only at the root node or throughout the branch & bound tree.

Our computational results consider four algorithms, for each general and security setting. The most interesting variation studied is based in normalization of Benders decomposition, whose objective is to reduce the number of feasibility and optimality cuts, and replace them for just one type of cut that we called *normalized cut*. We find that cut & branch, with variables in $\mathbb{R}$ in master problem and stabilization, outperforms the other methods considered in all instances. We suggest opportunities to improve these decomposition strategies in future research.

RESUMEN DE LA TESIS PARA OPTAR
AL GRADO DE: MAGÍSTER EN GESTIÓN DE OPERACIONES
POR: INGRID ALEJANDRA ARRIAGADA FRITZ
FECHA: 2021
PROF. GUÍA: FERNANDO ORDÓÑEZ PIZARRO

BENDERS DECOMPOSITION BASED ALGORITHMS FOR GENERAL AND SECURITY STACKELBERG GAMES

Los juegos de Stackelberg modelan situaciones donde existe más de un tomador de decisiones en una interaccion líder-seguidor, donde el líder toma una decisión en primer lugar y luego lo hace el seguidor. Cuando esto ocurre en un contexto de seguridad, con el líder actuando como defensor y el seguidor como atacante, se llama juego de seguridad de Stackelberg. Los juegos de Stackelberg son ampliamente usados para aplicaciones en economía, transportes, biología, finanzas, seguridad, entre otros.

Debido al uso de los juegos de Stackelberg generales y de seguridad en aplicaciones del mundo real, se ha vuelto importante resolver estos problemas en forma eficiente. De esta forma, los usuarios pueden tomar decisiones cuando lo requieran, cada día o cada hora. Esta necesidad de eficiencia ha producido diferentes formulaciones y métodos de solución. En esta tesis contribuimos al trabajo en formulaciones enteras mixtas de juegos de Stackelberg y algoritmos de descomposición para resolverlos. Específicamente, consideramos cortes de Benders generados de la formulación fuerte, y cortes de Benders normalizados como fue presentado por [29]. Por esta razón, esta tesis está enfocada en el desempeño de algoritmos.

La investigación reciente ha estudiado la eficiencia de diferentes formulaciones enteras mixtas para juegos de Stackelberg, incluyendo seguridad. Un hallazgo importante es que la formulación con el menor gap de integralidad, que considera más variables y restricciones, es la que tiene la relajación lineal más difícil de resolver. Este intercambio hace difícil decidir qué formulación es preferible para resolver instancias específicas, y sugiere fortalecer la formulación débil (FD) con cortes de la formulación fuerte (FF).

Este trabajo estudia el uso de cortes de Benders en la FD que son generados por la FF. En particular, investigamos el efecto de diferentes métodos de regularización, el uso de cortes de Benders normalizados, y la elección de si generar cortes solo en el nodo raíz o en todo el árbol branch & bound.

Nuestros resultados computacionales consideran cuatro algoritmos para cada tipo de juego de Stackelberg, general y seguridad. La variación más interesante que estudiamos está basada en la normalización de la descomposición de Benders, cuyo objetivo es reducir el numero de cortes de optimalidad y factibilidad, y reemplazarlos por solo un tipo de corte, al que llamamos *corte normalizado*. Concluimos que cut & branch, con variables en $\mathbb{R}$ en el problema maestro y estabilización, supera a los otros métodos considerados en todas las instancias. Se sugiere oportunidades para mejorar estas estrategias de descomposición en investigaciones futuras.

A mi Rodillo,
el ser mas incondicional que he conocido.
A mi madre y mi hermana,
mis pilares.
A Mitzi y Beatriz,
mis hermanas de otras madres.
A Carito, Barbarita y Javierita.
A mi Roberto.
A mi Perla querida.

♡

Acknowledgements

I want to thank my family for being my unconditional support during this whole process. Especially my mom Rosa, for being my greatest example of resilience and hard work; and my sister Belia, for being the sunshine of my life since I can remember. Also my nieces, they are like sisters to me. My sister Johanna, for her support and company during the last 7 years. My cousins, aunts and uncles, for supporting me in various ways. My brother-in-law Aliro, who has been a big support. My aunt Olguita, who left this world recently, that impacted my life in a priceless way and was present in all my student phases, especially this one.

I want to thank my friends Mitzi and Beatriz, who have always been with me, especially in my most difficult moments, supporting me in various ways in this and many other processes in my life. Also, my friends from Santiago: Chebote, Claudio, Zafrada, and very specially my bro Andres and his family. Thanks for your company and support while I was in town all alone.

My very special gratitude to my dear professor and friend Constanza Fosco, one of the most extraordinary people I ever met in the academic world. Thanks for supporting me during these 11 years, for her special help in the program requirement level adaptation, for her help in the application, her emotional support and continuing interest in my projects and well-being.

I am eternally grateful to my professor Fernando Ordóñez for the trust placed in me, and his inexhaustible patience and dedication since the first day. Also, I want to thank to professor Martine Labbé, who was always very interested in my work, and actively participated with ideas and suggestions. Thanks to professor Bernard Fortz, who contributed with ideas that are implemented in this thesis.

I want to thank to faculty members who helped me, my classmates and staff, specially Linda Valdés, Fernanda Melis, Tomás Lagos, Fernanda Jiménez and Constanza Contreras.

Finally, I want to thank Roberto, the one and only that has lived with me how my frustrations have changed to achievements during the long process of this thesis.

"Powered@NLHPC: This research/thesis was partially supported by the supercomputing infrastructure of the NLHPC (ECM-02)"

Agradecimientos

Quiero agradecer a mi familia, por ser mi apoyo incondicional durante todo este proceso. Especialmente mi madre Rosa, por ser mi mayor ejemplo de resiliencia y trabajo duro; y mi hermana Belia, por ser la luz de mi vida desde que tengo uso de razón. También a mis sobrinas, que son como hermanas para mí. A mi hermana Johanna, por su apoyo y compañía durante los últimos 7 años. A mis primas y tíos, que me han apoyado de diferentes formas. A mi cuñado Aliro, que ha sido siempre un gran apoyo. A mi tía Olguita, quien partió de este mundo recientemente, que impactó mi vida de una forma invaluable y estuvo presente en todas mis etapas estudiantiles, especialmente en esta.

Quiero agradecer a mis amigas Mitzi y Beatriz, quienes han estado siempre, especialmente en mis momentos más difíciles, apoyándome de diferentes formas en este y en muchos otros procesos de mi vida. A mis amigos de Santiago: Chebote, Claudio, Zafrada, y muy especialmente mi bro Andrés y su familia. Gracias por su compañía y apoyo mientras me encontraba en Santiago completamente sola.

Un agradecimiento muy especial a mi querida profesora y amiga Constanza Fosco, una de las personas más extraordinarias que he conocido en el mundo académico. Gracias por apoyarme durante estos 11 años, por su especial ayuda para adaptarme al nivel de exigencia del programa, por su ayuda en la postulación, su apoyo emocional y su constante interés en mis proyectos y mi bienestar.

Estoy eternamente agradecida de mi profesor guía Fernando Ordóñez, por toda la confianza depositada en mí, por su inagotable paciencia y dedicación desde el primer día. También quiero agradecer a la profesora Martine Labbé, que siempre estuvo muy interesada en mi trabajo y participó activamente con ideas y sugerencias. Al profesor Bernard Fortz, quién contribuyó con ideas que también están implementadas en esta tesis.

También quiero agradecer a las personas de la facultad que me ayudaron, a mis compañeros y también el personal, especialmente Linda Valdés, Fernanda Melis, Tomás Lagos, Fernanda Jiménez y Constanza Contreras.

Por último, quiero agradecer a Roberto, el único que ha vivido conmigo cómo mis frustraciones se han convertido en logros en el largo proceso de esta tesis.

"Powered@NLHPC: Esta investigación/tesis fue parcialmente apoyada por la infraestructura de supercómputo del NLHPC (ECM-02)".

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The world has billions of inhabitants who interact with each other. For that reason, there are innumerable situations that imply the interaction between different decision makers. Many of that interactions consider the dependence of the results with the set of decisions of all the involved agents. For example, in soccer, when a player kicks a penalty, the result doesn't depend only on where the player decides to kick, but on where the goalkeeper decides to jump too. There exist many other examples for this kind of situations, and they are studied by *Game Theory*.

There exist some situations where the relationship between decision makers is a leader-follower interaction, where leader decides first, and follower decides in function of the first. This kind of specific game theory situations are called *Stackelberg Games*. Stackelberg games are widely used for applications in economics, transportation, biology, security, among many others.

We are interested in Stackelberg security games, where leader takes a defensive role, and follower takes a offensive role. Leader has limited defensive resources for a bigger number of targets to protect, and must decide which of them will cover, and follower must choose which target to attack. There exist many real applications of this methodology, for example: PROTECT, a system that the United States Coast Guard uses to schedule patrols in the Port of Boston, [1]; and ARMOR, an assistant deployed in Los Ángeles International Airport to randomize checkpoints on the roadways entering the airport and canine patrol routes within the airport terminals.

Stackelberg security games are modeled using mixed integer linear programs. This kind of programs are naturally challenging to solve, and get harder when facing large scale problems. Let's consider the problem of protecting the Chilean ground border, that is almost 8000 km of distance. There exist many people that are interested in illegal immigration, that means many possible attackers to face. If we think about the probabilities of face an specific attacker (immigrant) in a specific target (zone of the border), they are negligible. The problem of choosing which zones of the border protect everyday is a large scale problem, and should be very difficult to solve. This could take days, months or even years of computational processing time.

In real world, we need applications that solve this kind of problems in practical time, that may be used in everyday decisions. There is the importance of investigating algorithms that help to improve these applications. For the detailed reasons, this work has an algorithmic focus, where we introduce variations in the work of [19] for trying to get improvements of performance.

This thesis is structured with the following chapters:

In chapter 2, we show a bibliographical review of the main fields of study involved: game theory, Stackelberg competition, bilevel programming, Benders decomposition, mixed integer linear problems, and its derivatives.

In chapter 3, we formally define general and security Stackelberg problems.

In chapter 4, we show a work developed in [20], where they took existing general and security Stackelberg games formulations and transform them using programming techniques (projections, change of variables, change of constraints, Fourier-Motzkin elimination, reformulation-linearization technique). This allows to get formulations with different advantages, which makes possible to create algorithms that mix formulations in order to enhance performance.

In chapter 5, we workout different Benders decomposition approaches to be compared in this thesis. We introduce a special type of decomposition based on [29], where we use only one type of cut, which is normalized.

In chapter 6, we explain the branch & cut and cut & branch algorithms that use the Benders decomposition approaches in chapter 5, due to [19]. These algorithms use the linear relaxations of the fast and weak formulations from chapter 4, and strengthen them using the Benders cut from tight formulations. In the same chapter, we briefly describe some heuristics proposed by [19] for enhancing the algorithms performance.

In chapter 7, we run experiments in order to decide the best heuristic configurations for our algorithms, and compare them performance against [19] algorithms.

Chapter 2

Theoretical framework

In this thesis, we present Benders decomposition to solve mixed integer optimization formulations of Stackelberg games. As such, the review of relevant literature presented in this chapter touches on game theory and Stackelberg games models, and significant solution algorithms for Stackelgerg games, that is: algorithms for bilevel optimization, Benders decomposition and mixed integer optimization solution methods.

2.1 Game theory

Briefly speaking, game theory is the mathematical study of distributed decision making where the individual has to take other individuals' decisions into account. The reward for an agent depends of the set of decisions of all the involved agents.

In game theory, every game includes the following concepts:

Players: individuals that make decisions with the objective of maximize their utility.

Strategy: set of actions players can select to do.

Payoffs: set of results of the game considering every possible simultaneous strategies of the players.

2.1.1 History

In the XVIII century there were many contributions that could be considered as precursors of game theory and led to the formal mathematical study that set the theoretical basis of this science, [54] present a review of these. In the next century, [24] and [12] developed their respective duopoly models, both of them using the Nash equilibrium concept.

It can be considered that game theory was formally launched by Zermelo's theorem [64], it was related to the game of chess, but applicable to any kind of game in which two opponents play against each other with the exclusion of chance events. The work by [49] provides a translation of this theorem. This is the first formal mathematical theorem in this field.

Zermelo's theorem states: *"Either white can force a win, or black can force a win, or both sides can force at least a draw"*.

In the decade of 20s, [55] proves a theorem known as *Minimax Theorem*. This theorem states: *"Every finite, zero-sum, two person game, has optimal mixed strategies that maximize potential gains or minimize potential losses"*. In the next decade, [57] launched Stackelberg competition.

Most authors consider [56] as the starting point of game theory as a science, because of the huge impact of this work in the discipline. In this book, all the important concepts in game theory are formally stated: game, strategy, payoff, etc.

In the subsequent decades, research in game theory science exploded. Some of these works are: [43], where non-cooperative games basis were set and Nash equilibrium concept was formally defined; [37], where extensive games formulation was introduced; [2], where the strong equilibrium concept was introduced. Other important fact is that Game theory began to be applied in different fields, for example: political sciences [50] and evolutive biology [41]. Game theory was very important even during the Cold War in the 60s, in [3] there is a collection of articles that were developed for a United States Agency related to weapons control.

In the last decades, research in game theory has received multiple awards and recognitions, including the Nobel Prize (John Nash, 1994; Robert Aumann 2005; among others).

2.1.2 Games representation

There are two forms to represent a game: normal and extensive form.

Normal form games are represented by a payoffs matrix. The players have a finite number of actions, and have to choose one of them. In the case of two players (A and B), player A's actions are represented by rows and player B's actions by columns respectively. When both players actions are selected, their respective utilities are represented by the values at the intersection of the row and column vectors of each decision.

A generic example of a normal form game is presented in table (2.1), the rows actions represent player A, and column actions represent player B. In this game, if player A takes action 1 and player B takes action 1, the utilities are 5 and 4 respectively. The utilities are inverted if both players take action 2. On the other hand, if player A takes action 2 and player B takes action 1, the utility is 0 for both players, and if we invert the actions, the utility for both players is 1.

	Action 1	Action 2
Action 1	(5,4)	(1,1)
Action 2	(0,0)	(4,5)

Table 2.1: Normal form game representation.

Extensive form games model situations in which there is some order between players

actions. This games are represented by a decision tree that shows all the possible decisions and utilities.

Figure (2.1) shows a simple example of an extensive form game, heads or tails game. The tree shows that if player A gets *"head"* and player B gets the same, player A has to pay 1 unity to player B, but if player B gets *"tail"*, player B has to pay 1 unity to player A. On the other hand, if player A gets *"tail"* and player B gets *"head"*, player B has to pay 1 unity to player A, but if both players get *"tail"*, player A has to pay 1 unity to player B.

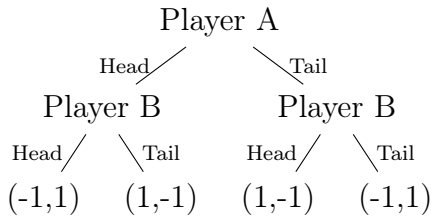

Figure 2.1: Extensive form game representation.

There are two types of strategies that players do in game theory: pure and mixed strategies. A pure strategy is the selection of a single action of a set of possible actions. A mixed strategy is a probability distribution over the set of actions. A mixed strategy gives the likelihood that each action is selected by the player. A pure strategy is a particular case of a mixed strategy, where one of the options has a probability of 1, and the remaining options have a probability of 0.

2.1.3 Examples of games

In this section, we briefly describe a few famous examples of games.

The prisoner's dilemma

This game models the situation in which two criminals are arrested by the police, but police doesn't have enough evidence to convict them, so they need a confession of the crime.

Officials separate the prisoners to question them so they can't communicate with each other. There are different possible results from the interrogation:

1. If both confess, they can be charged for the serious crime.
2. If nobody confesses, they can only be charged for a minor crime.
3. If one confesses and the other one doesn't, the person that confesses has his punishment reduced while the other one gets charged in addition with obstruction.

The most favorable action for two criminals is to not confess. The problem is that there is no certainty about the other prisoner's decision, and the risk of not confessing with the other confessing is too big. So, the rational outcome is that both prisoners will confess and are charged for the crime.

Rock-paper-scissors

This is a well known simultaneous game where the players have to choose between three options: rock, paper or scissors.

The rules of this game are: rock defeats scissors, paper defeats rock and scissors defeats paper. If both players choose the same strategy, the result is a draw.

Cournot competition

In this game, there is a fixed number of firms that produce an homogeneous product.

The firms have to decide the quantity to produce simultaneously, and seek to maximize profit. The price is defined for a demand function that is decreasing of the total quantity produced by the market.

Volunteer's dilemma

This game models a situation in which there are a set of players. Each player has two options: make a sacrifice to benefit everybody or wait for someone else's sacrifice:

1. If at least one of player sacrifices, sacrificed players get 0 utility and not sacrificed players get a utility of 1.
2. If nobody sacrifices, everybody loses 15.

These are a few classic games, and there are several games in different fields of study, see for example [38] and [48]. In this thesis, we work in Stackelberg games, which are described in the next section.

2.2 Stackelberg competition

Stackelberg competition, was introduced by [57], to model a strategic game between two firms, the leader and the follower. The leader firm decides first the quantity to produce in order to maximize profits, and the follower, after observing the leader's decision, chooses a production level with the same objective. Of course, the production level chosen by the follower is the best response to the leader's decision.

The leader knows that the follower is observing his behavior, and makes his decisions taking this in account.

Stackelberg games are a model of imperfect competition (because of the market power of leader), non-cooperative and sequential.

We can compare Stackelberg competition to Cournot and Bertrand competition.

Cournot competition, as described in §2.1.3, is a model where firms decide simultaneously the quantity to produce of an homogeneous product. The big difference with Stackelberg competition is that the second is sequential.

Bertrand competition is a non-cooperative model, with imperfect information and simultaneous, just as Cournot competition. The difference between is that Cournot compites by deciding quantities, and Bertrand compites deciding prices.

Research on the Stackelberg game concept has extended its use to model interactions in several fields where there is a leader-follower relationship [21]. For example, in telecommunications [15], [60]; transportation [38], [18]; inventory [63], [34]; among many others.

Stackelberg competition is represented throughout this thesis as normal form game (defined in §2.1.2) and played with mixed strategies.

Let's consider a payoff bimatrix similar to (2.1), with a player represented in the rows, and the other player in the columns. Let I and J denote the set of strategies for the row and column players respectively, and let (R, C) be the bimatrix where $R, C \in \mathbb{R}^{|I|\times|J|}$. Consider the $|I|$ and $|J|$ dimensional simplices:

$$\mathbb{S}^{|I|} = \left\{ x \in [0,1]^{|I|} : \sum_{\mathrm{i}\in I} x_{\mathrm{i}} = 1 \right\}, \tag{2.1}$$

$$\mathbb{S}^{|J|} = \left\{ q \in [0,1]^{|J|} : \sum_{j\in J} q_j = 1 \right\}, \tag{2.2}$$

where $x \in \mathbb{S}^{|I|}$ is a mixed strategy for the row player, where the *i-th* strategy is played with probability x_{i}, and $q \in \mathbb{S}^{|J|}$ is a mixed strategy for the column player, where the *j-th* strategy is played with probability q_j.

2.2.1 Stackelberg equilibrium

The standard solution concept in game theory is *Nash equilibrium*: a strategy profile such that no player can gain by unilaterally deviating to another strategy [36]. This notion captures a *steady state* of the play of a strategic game in which each player holds the correct expectation about the other players' behavior and acts rationally [45].

The solution concept used throughout this thesis is *Stackelberg equilibrium*. This is the solution concept of choice when the game is sequential instead of simultaneous.

According to [36], Stackelberg equilibrium is a refinement of Nash equilibrium specific to Stackelberg games. It is a form of subgame perfect equilibrium, in which each player chooses a best response in any subgame of the original (where subgames correspond to partial sequences of actions).

There are two types of Stackelberg equilibria in [40], typically called *strong* and *weak* equilibrium. The strong form assumes that the follower will always choose the optimal strategy for the leader in cases of multiple best responses, while the weak form assumes the opposite, the follower will choose from the multiple best responses, the worst strategy for the leader. A strong Stackelberg game equilibrium exists in all Stackelberg games, but a weak

Stackelberg equilibrium may not [8]. Additionally, according to [59], leader can often induce the favorable strong equilibrium by selecting a strategy arbitrarily close to the equilibrium that causes follower to strictly prefer the desired strategy. Strong equilibrium is also the most commonly adopted concept in related literature [36]. For these three reasons, the equilibrium type used throughout this thesis is the strong Stackelberg equilibrium.

Strong Stackelberg equilibrium

Let x be a leader strategy, $q(x)$ be the follower best response strategy, U_{leader} the leader's utility and $U_{follower}$ the follower's utility. According to [62], a profile of mixed strategies $(x, q(x))$ forms a strong Stackelberg equilibrium if they satisfy the following:

1. The leader plays a strategy that maximizes its utility:

$$U_{leader}(x, q(x)) \geq U_{leader}(x^{'}, q(x^{'})), \quad \forall x^{'} \tag{2.3}$$

   i.e., leader's strategy maximizes his utility which is bigger than with any other strategy.

2. The follower plays a best response:

$$U_{follower}(x, q(x)) \geq U_{follower}(x, q^{'}), \quad \forall x, q^{'} \tag{2.4}$$

   i.e., follower's strategy maximizes his utility which is bigger than with any other strategy.

3. The follower breaks ties optimally for the leader:

$$U_{leader}(x, q(x)) \geq U_{leader}(x, \tau), \quad \forall \tau \in S(x), \tag{2.5}$$

   where $\mathcal{S}(x)$ is the set of follower best responses to x, i.e., if follower has different strategies that maximize his utility, he chooses the one which maximizes leader's utility.

2.2.2 Types of games

Types of Stackelberg games addressed throughout this thesis are two: general and security Stackelberg games. Brief definitions of them are provided in this section.

General Stackelberg games

General Stackelberg games (GSG) can model any adversarial situation between players with different objectives, each striving to optimize a certain payoff in a sequential, one-off encounter. One of the players can commit to a given action first and he is referred to as the leader, whereas the other player, which responds to the leader's action, is referred to as the follower [19].

Security Stackelberg games

A security Stackelberg game (SSG) is a specific case of a GSG where the pure strategies for the leader, now referred to as the defender, involve allocating a limited number of security resources (law enforcement officers, for example) to protect a subset of targets (critical infrastructure, for example) and the pure strategies for each follower type consist in attacking a single target [19].

This means that for Security Stackelberg games there is a structure on the payoffs, where a reward or penalty for each player only depends on whether the attacked target is patrolled or not. That is if i $\in I$ represents the patrolling strategy i and $j \in J$ represents the strategy of attacking target j, then

$$R_{\mathrm{i}j} = \begin{cases} R_j^D & \text{if } j \in \mathrm{i} \\ P_j^D & \text{o/w} \end{cases} \quad \text{and} \quad C_{\mathrm{i}j} = \begin{cases} P_j^A & \text{if } j \in \mathrm{i} \\ R_j^A & \text{o/w} \end{cases}$$

Here $R_j^D(P_j^A)$ represents the reward for the defender (penalty of the attacker) of an attack on a patrolled target j ($j \in$ i) and P_j^D (R_j^A) stands for the penalty for the defender (reward for the attacker) of an attack on an unprotected target j ($j \notin$ i).

2.3 Bilevel programming

Stackelberg games are an example of bilevel programming problems.

Bilevel programming refers to mathematical optimization problems that contains an optimization problem in the constraints. This kind of problems was introduced as a field by [17], with influences from [58].

The contribution of [58] to this field comes from the problems involving more than one player with conflicted positions and sequential decisions, which implies opposite optimization problems, which could be approached with this kind of optimization programs.

Bilevel programs are specially useful when a situation of sequential decision is faced, and the decision order is important. This is the case of Stackelberg games, where the leader makes decisions before the follower.

The structure of these types of programs is hierarchical, with a nested optimization problem that established constraints on a variable of the main decision problem. First level problem (or main decision problem), represented by the main objective function and constraints, constitutes the leader level, while the second level problem (or nested problem), which is represented by the nested objective function and constraints, constitutes the follower level.

According to [6], a bilevel optimization program has the following structure:

$$\begin{aligned} &\min_x \quad F(x, y(x)) \\ &\text{s.t} \quad x \in X = \{x : H(x) \geq 0\}, \\ &\qquad \min_y \quad f(x, y) \\ &\qquad \text{s.t} \quad g(x, y) \geq 0, \\ &\qquad\qquad y \in Y = \{y : G(y) \geq 0\}, \end{aligned} \tag{2.6}$$

where $F(x, y(x))$ represents the leader objective function, $H(x)$ and $G(y)$ are leader's and follower's constraint functions respectively, and $g(x, y)$ represents follower's constraint functions given leader's decision.

Leader minimizes F over a feasible x, anticipating follower's best response $y(x)$ given x, obtained by optimizing the second level problem. Thereby, optimal decision vector returned by bilevel programming are a pair of mutual best responses [19].

Bilevel linear programming problems (BLPP) are bilevel problems involving only linear objective and constraint functions. [10] and [35] show that solving BLPP is NP-hard.

According to [52], there are four main categories of solution methods based on non-linear optimization to solve bilevel problems:

1. *Single level reduction:* includes reformulation techniques which transform the bilevel program into a single level problem.
2. *Descent methods:* includes algorithms which extract gradient information from the lower problem and use it to compute directional derivatives for the leader's objective function.
3. *Penalty function methods:* includes penalty algorithms, which introduce a term that incorporates a penalty associated with the violation of certain optimality conditions, leading this way the search towards the optimal solution.
4. *Trust region methods:* includes algorithms that approximate a certain region of the objective function with a model function. The region is expanded if approximation is good, and contracted otherwise.

Throughout this thesis, we use the same approach of [46], which is based in methods of single level reduction, that leads to the so-called *mathematical problems with complementarity constraints (MPCC)*.

2.3.1 Mathematical problems with complementarity constraints

Bilevel programming problems are often reformulated using the Karush Kuhn Tucker (KKT) conditions for the lower level problem resulting in MPCC [26].

Recall that KKT conditions are: Lagrangian aligned gradients, primal feasibility, dual feasibility and complementary slackness [16].

The main idea is getting a single level problem, by transforming the whole nested problem into KKT optimality conditions, i.e., a set of constraints that characterize the optimal solution of the nested problem.

In general bilevel problems, this kind of transformation may not lead to a tractable single level optimization problem, because the KKT constraint on the aligned gradients can lead to non-convexity. In addition, the complementarity constraints are in general non-convex . But in the case of BLPP, aligned gradients constraint is linear [52].

In our Stackelberg games approach, the follower's problem is linear. Therefore, there is not difficulty in handling the constraint on the aligned gradients. To handle the complementarity constraints that arise when applying KKT conditions, we use an approach presented in [46]. This procedure will be detailed in §4.2.1 and §4.3.1.

2.4 Benders decomposition methods

A linear optimization problem is known as large scale problem when it has a huge number or variables or constraints. Large scale optimization problems are computationally challenging and often not solvable within reasonable computational time or memory limits [7].

To address this complication, there exist decomposition methods that could allow to solve some large scale problems in reasonable time. The main idea of the algorithms is to work with a subset of the variables or constraints of the original problem, and iteratively increase this subset if its needed.

There exist several kind of decomposition algorithms (for example: delayed column generation, delayed constraint generation or cutting plane method, Dantzig-Wolfe algorithm), all of them exploiting different properties of the problems to solve.

In this thesis, we use the Benders algorithm [11]. This algorithm is described in §2.4.1.

2.4.1 Benders decomposition algorithm

Consider the generic MILP problem of the form:

$$\begin{aligned} \min_{x,\,y} \quad & c^T x + \mathrm{d}^T y \\ \text{s.t} \quad & Gx + Hy = g, \quad \text{both variables} \\ & Ax \succeq a, \quad \text{only integer variables} \\ & By \succeq b, \quad \text{only continuous variables} \\ & x \in \mathbb{Z}^n \\ & y \succeq 0 \end{aligned} \tag{2.7}$$

where $y \in \mathbb{R}^r$, G and H have m rows, A has p rows, and B has q rows.

As can be seen, the problem has three kind of constraints: the first set of constraints involves both types of variables, integer and continuous; the second set only involves integer variables; and the third involves only continuous variables. This fact is very important to understand the algorithm description in the following lines.

The work in [11] proposes to divide the problem (2.7) in two other problems: the *master problem* and the *subproblem*. These problems are as follows:

Master problem

$$\begin{aligned} \min_{x} \quad & c^T x + \theta \\ \text{s.t} \quad & Ax \succeq a, \\ & (g - Gx)^T \alpha + b^T \beta \leq \theta, \quad \forall (\alpha, \beta) \in U \\ & (g - Gx)^T \alpha + b^T \beta \leq 0, \quad \forall (\gamma, \delta) \in V \\ & x \in \mathbb{Z}^n, \end{aligned} \tag{2.8}$$

where U and V are the sets of extreme points and extreme rays of the feasible region of *dual subproblem* (2.10). These sets can start out as empty sets.

Subproblem

$$\begin{aligned} Q(x) = \min_{y} \quad & \mathrm{d}^T y \\ \text{s.t} \quad & Hy = g - Gx \quad (\alpha) \\ & By \succeq b \quad (\beta) \\ & y \succeq 0 \end{aligned} \tag{2.9}$$

The *master problem* (2.8) has only the integer variables x in the objective function and the constraints, in addition, has the θ variable that takes into account the objective function of the *subproblem*.

The *subproblem* (2.9) is defined for every vector x and has in the objective function only the continuous variables, and only continuous constraints (the first constraint is not mixed but continuous, because the x variables are parameters of this problem, comming from the *master problem*).

Separation problem

This is the dual of subproblem:

$$\begin{aligned} \max_{\alpha,\beta} \quad & (g - Gx)^T \alpha + b^T \beta \\ \text{s.t} \quad & H^T \alpha + B^T \beta \preceq \mathrm{d}, \\ & \beta \succeq 0 \end{aligned} \tag{2.10}$$

In a large scale optimization problem, master problem has a huge number of possible constraints, one constraint for every extreme point and extreme ray of the polyhedron in (2.10), so we need to apply a method to solve it using a subset of these constraints. The Benders algorithm uses cutting planes methods to add these constraints as needed.

We now outline the algorithm according to [13].

Algorithm

1. An iteration starts by solving the *relaxed master problem* (2.8) in which only a subset of the constraints are included. Let the optimal solution vector be (x^*, θ^*).
2. We solve the *separation problem* (2.10).
3. If the *subproblem* is feasible and the optimal cost is less than or equal to θ^* or a defined tolerance, i.e., $(g - Gx^*)^T \alpha^* + b^T \beta^* \leq \theta^*$, it means that all constraints are satisfied, we have an optimal solution to the *master problem* and the algorithm terminates. (x^*, θ^*) is the optimal solution of the problem.

4. If the *subproblem* is feasible and the optimal cost is greater than θ^*, i.e., $(g - Gx^*)^T\alpha^* + b^T\beta^* > \theta^*$, an optimal basic feasible solution (α^*, β^*) to the *dual subproblem* (2.10) is identified, and we add the following constraint to the *relaxed master problem*:

$$(g - Gx)^T\alpha^* + b^T\beta^* \leq \theta \tag{2.11}$$

5. If the *subproblem* is infeasible or the *dual subproblem* is unbounded, we identify an extreme ray (γ^*, δ^*) of (2.10), and we add the following constraint to the *relaxed master problem*:

$$(g - Gx)^T\gamma^* + b^T\delta^* \leq 0 \tag{2.12}$$

6. Repeat until the algorithm terminates.

The main complication of this algorithm arises in the solution method of relaxed master problem, which includes integer variables. That complication is addressed in §2.5.

2.4.2 Minimally infeasible subsystems

While running Benders decomposition algorithm, some variations can be done looking to improve its performance. In this thesis, we will work with an approach based in Farkas' Lemma [27], as described in [31] and motivated by the work of [29] and [30].

Let's consider a system of linear inequalities:

$$Ax \preceq b \tag{2.13}$$

where A is a $m \times n$ matrix and b a m-vector. Let us suppose that (2.13) is infeasible. A *minimally infeasible subsystem* of (2.13) is a subsystem of infeasible inequalities that can be made feasible by dropping just one of them from the system. Hence, those subsystems are the smallest possible infeasible polyhedra of the system.

Recall that showing the infeasibility of a system is equivalent to showing that its dual system is unbounded:

$$\begin{aligned} \min_{y} \quad & y^T b \\ \text{s.t} \quad & y^T A = 0, \\ & y \succeq 0 \end{aligned} \tag{2.14}$$

[31] prove that finding the extreme rays of (2.14) is equivalent to finding the extreme points of the alternative polyhedron:

$$\begin{aligned} & y^T A = 0 \\ & y^T b \leq -1, \\ & y \succeq 0 \end{aligned} \tag{2.15}$$

Even further, the rows of any minimally infeasible subsystem of (2.13) are indexed by the support vertices of (2.15).

By using this fact, we can do Benders cut generation dispensing with finding extreme rays, and we can just find extreme points of an adapted polyhedron. The application of this to our approach is detailed in §5.2.2 and §5.3.2, for general and security Stackelberg games respectively.

2.5 Mixed integer linear problems

A mixed integer linear problem (MILP) is a problem type in which at least one of the variables is integer, and objective and constraint functions are linear.

Mixed integer linear problem structure looks as follows:

$$\begin{aligned} \min_{x,\,y} \quad & c^T x + g^T y \\ \text{s.t} \quad & Ax + By \leq b, \\ & x \in \mathbb{R}^n_+, \\ & y \in \mathbb{Z}^p_+, \end{aligned} \tag{2.16}$$

here, x is an n-vector of variables greater than or equal to 0 and y is a p-vector of integer variables greater than 0. The problem has a mix of integer and continuous variables.

MILPs are present in many fields and disciplines, for that reason, several authors have been interested in their study. For example, [14], [44] and [61], among many others.

Solving MILPs could be a very difficult task [33], much more difficult than LPs. In particular, solution methods for MILP use LP solvers as subroutines.

Most of solution methods for MILPs start with relaxation of the problems, i.e., change the original problem for an easier to solve problem, in order to find bounds. For example, a *linear relaxation*, that consists in considering integer variables as continuous variables. If we are lucky and the solution to the relaxed problem is feasible for the MILP, then problem is solved. But if the solution is not integer, this relaxation (for a minimization problem) gives a lower bound of MILP.

There are several types of solution methods that make use of this relation between MILP and the relaxed version, for example: *cutting plane methods*, *enumerative methods*, *hybrid methods*, and others.

In solving integer problems, cutting plane method is an iterative algorithm that successively tightens the relaxation of the problem by adding constraints that result valid for all integer solutions. To do that, this method uses duality theory. It was first introduced by [32]. There exist several kind of cutting plane methods: Gomory's cutting plane, Benders decomposition (defined in §2.4.1), disjuntive cuts [4], among others.

Enumerative algorithms to solve mixed integer problems are exact. They explore feasible solutions of the problem using a tree of possibilities, but applying rules that allow to identify parts of the tree that do not need to be explored, either because the problem presents some exploitable symmetry or because it can be shown that the optimal solution is not in those zones. Some examples of enumerative methods are: branch & bound [39], backtracking search, among others.

Hybrid methods mix cutting plane and enumerative methods. Some well known hybrid methods are branch & cut [5] and cut & branch.

In this thesis, the used solution methods are hybrid, specifically branch & cut and cut & branch. These are based in branch & bound method.

Branch & bound

This method was first proposed in [39], where it was briefly defined as "simple numerical algorithm for the solution of programming problems in which some or all variables can take only discrete values"

The branch & bound algorithm first solves the linear relaxation of the problem, i.e., the problem without integrality constraints, and gets a lower bound (when minimizing). Then it iteratively adds constraints that fix variables in different integer values, forming scenarios for each of them. If all possible scenarios have been considered, the best lower bound is the optimal value of the problem, because has been proved that there is no integer variable solution with better result. An important remark is that all possible scenarios are considered, but not necessarily solved, because most of them are discarded using some logical rules.

Consider the generic MILP problem (2.7). This problem has integer and linear variables. We now define his linear relaxation:

Linear relaxation

$$\begin{aligned} \min_{x,\,y} \quad & c^T x + \mathrm{d}^T y \\ \text{s.t} \quad & Gx + Hy = g, \\ & Ax \succeq a, \\ & By \succeq b, \\ & x \succeq 0, \\ & y \succeq 0. \end{aligned} \tag{2.17}$$

where x and y are linear variables with positive values.

We now outline the algorithm.

Algorithm

1. The root node of the tree is the solution of the linear relaxation (2.17), call it (x^0, y^0). If the solution satisfies the constraints of (2.7) (integer variables), then this is the problem

solution, and algorithm terminates.

2. If (x^0, y^0) is not the required solution it means that some integer variable is fractional, say x_1^0. Then two edges are drawn, each of them adding a different constraint to the original problem. These constraints impose the upper and lower integer bound on the variable that is fractional. That is for each problem we add one of the following constraints:

$$x_1 \leq \lfloor x_1^0 \rfloor \tag{2.18}$$

$$x_1 \geq \lceil x_1^0 \rceil \tag{2.19}$$

We now have two new problems. First problem with one new constraint (P1):

$$\begin{aligned} \min_{x,\,y} \quad & c^T x + \mathrm{d}^T y \\ \text{s.t} \quad & Gx + Hy = g, \\ & Ax \succeq a, \\ & By \succeq b, \\ & x \succeq 0 \\ & y \succeq 0, \\ & x_1 \leq \lfloor x_1^0 \rfloor \end{aligned} \tag{2.20}$$

Second problem with the other constraint (P2):

$$\begin{aligned} \min_{x,\,y} \quad & c^T x + \mathrm{d}^T y \\ \text{s.t} \quad & Gx + Hy = g, \\ & Ax \succeq a, \\ & By \succeq b, \\ & x \succeq 0 \\ & y \succeq 0, \\ & x_1 \geq \lceil x_1^0 \rceil \end{aligned} \tag{2.21}$$

These constraints force the optimal solutions avoid the fractional region where the original problem was optimal.

3. Solve two new problems. If one of the solutions satisfies the constraints of (2.7) (integer variables), it can be saved as a possible problem solution and that branch is not explored anymore. If we get infeasible problem, that branch is discarded.
4. If there is no problem solution or infeasibility, repeat from step 2, now in (2.20) and (2.21), until finding possible solutions or discarding in all branches. The algorithm terminates.
5. If we get more than one possible solution, the problem solution is the best of them. If we don't get any possible (integer) solution, the problem is infeasible.

Example

Figure (2.2) shows branch & bound scheme for an example maximization problem in [53]:

$$\begin{aligned} \max_{x_1,\, x_2} \quad & 100x_1 + 150x_2 \\ \text{s.t} \quad & 8,000x_1 + 4,000x_2 \leq 40,000, \\ & 15x_1 + 30x_2 \leq 200, \\ & (x_1, x_2) \in \mathbb{Z}_+^2 \end{aligned} \tag{2.22}$$

This problem has two integer variables x_1 and x_2, and can be seen that the root node linearly relaxed optimal solution is 1055.56, with $x_1 = 2.22$ and $x_2 = 5.56$, this solution is infeasible to integrality constraints, so it was necessary to branch the problem.

In level 2 of the tree, P_1 has optimal value of 1000 and $x_1 = 2.5$ and $x_2 = 5$, P_2 has optimal value of 1033 and $x_1 = 1.33$ and $x_2 = 6$. Integer solution is not reached and node with max optimal value is P_2, so that node is branched.

In level 2 of the tree, $P_{2,1}$ has optimal value of 1025.5 and $x_1 = 1$ and $x_2 = 6.17$, $P_{2,2}$ is infeasible. Integer solution is not reached, so we branch the one feasible node $P_{2,1}$.

In level 3, $P_{2,1,1}$ has optimal value of 1,000 and $x_1 = 1$ and $x_2 = 6$, $P_{2,1,2}$ is infeasible. Integer solution is reached at node $P_{2,1,1}$. So, the final optimal solution is $(x_1, x_2) = (1, 6)$ and optimal value is $v^{2,1,2} = 1,000$.

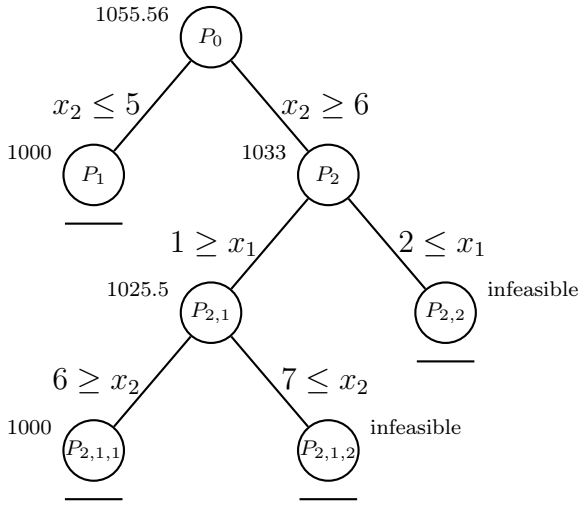

Figure 2.2: Branch & bound algorithm scheme.

2.5.1 Branch & cut

As explained, branch & cut is an hybrid method. This algorithm mixes a cutting plane and branch & bound method.

First main topic of this thesis is solving large scale MILP problems, using a branch & cut algorithm with cuts generated from Benders decomposition. Benders decomposition algorithm is used to address the large scale feature of the problem, while branch & bound is used to manage integrality. Benders decomposition algorithm was described in §2.4.1.

Branch & cut algorithm works in the branch & bound scheme, but adding extra cuts (if necessary) in each node by using cutting planes methods, specifically in the work presented here we use Benders decomposition method.

2.5.2 Cut & branch

Second main topic of this thesis is solving the same large scale MILP problems, but using cut & branch algorithm. This algorithm works similar to branch & cut, but only adds cuts in the root node. Remaining nodes just add the branching constraints.

Chapter 3

Problem definition

In this chapter, we formally define Bayesian general Stackelberg games (GSGs) and their extension to the security field, Bayesian security Stackelberg games (SSGs). Bayesian Stackelberg games are those where the leader may face one of multiple follower types.

3.1 General Stackelberg games

We now define the sets, parameters and variables of Bayesian GSGs.

Sets:

K: set of followers.

I: set of leader pure strategies.

J: set of follower pure strategies.

Parameters:

$\pi^k \in [0,1]$: probability of facing follower $k \in K$.

$R^k \in \mathbb{R}^{|I|\times|J|}$: leader's reward matrix when facing follower $k \in K$.

$C^k \in \mathbb{R}^{|I|\times|J|}$: reward matrix for follower $k \in K$.

Variables:

$x_{\mathrm{i}} \in [0,1]$: probability for leader to play pure strategy $\mathrm{i} \in I$, such that $\sum_{\mathrm{i}\in I} x_{\mathrm{i}} = 1$.

$q_j^k \in \{0,1\}$: decision of follower $k \in K$ to play pure strategy $j \in J$, such that $\sum_{j\in J} q_j^k = 1$.

The expected reward for the leader is:

$$\sum_{\mathrm{i}\in I}\sum_{j\in J}\sum_{k\in K}\pi^k R^k_{\mathrm{i}j}x_{\mathrm{i}}q^k_j \tag{3.1}$$

The expected reward for the k follower is:

$$\sum_{\mathrm{i}\in I}\sum_{j\in J}C^k_{\mathrm{i}j}x_{\mathrm{i}}q^k_j \tag{3.2}$$

The following model, (BLP-G$_{x,q}$), is a bilevel programming formulation for the Bayesian general Stackelberg game problem:

(BLP-G$_{x,q}$)

$$\begin{aligned}
\max_{x,\,q}\quad & \sum_{\mathrm{i}\in I}\sum_{j\in J}\sum_{k\in K}\pi^k R^k_{\mathrm{i}j}x_{\mathrm{i}}q^k_j \\
\text{s.t}\quad & \sum_{\mathrm{i}\in I}x_{\mathrm{i}}=1, \\
& x_{\mathrm{i}}\in[0,1] \qquad \forall \mathrm{i}\in I, \\
& q^k\in \arg\max_{r^k}\left\{\sum_{\mathrm{i}\in I}\sum_{j\in J}C^k_{\mathrm{i}j}x_{\mathrm{i}}r^k_j\right\} \qquad \forall k\in K, \\
& \quad r^k_j\in[0,1] \qquad \forall j\in J,\forall k\in K, \\
& \quad \sum_{j\in J}r^k_j=1 \qquad \forall k\in K.
\end{aligned} \tag{3.3}$$

The objective function maximizes leader's expected reward. First and second constraints characterize the mixed strategies considered by the leader. Nested problem maximizes follower's reward by playing best response to the leader's decision.

3.2 Security Stackelberg games

A security Stackelberg game is a specific type of GSG where the leader is called defender, and follower is called attacker. In this case, the pure strategies for the defender involve allocating limited number of security resources to protect a subset of targets, and the pure strategies for each attacker consist in attacking a single target [36].

Let n be the number of targets, and let m be the number of security resources available to protect these targets, some $m < n$, that is the number of resources is lower than the number of targets. Allocating resource to some target protects that target from attack.

We now define the sets, parameters and variables of Bayesian SSGs.

Sets:

K: set of attackers.

I: set of defender pure strategies, composed of all $\sum_{\mathrm{i}=1}^{m} \binom{n}{\mathrm{i}}$ subsets of at most m targets that can be protected simultaneously.

J: set of attacker pure strategies.

Parameters:

$\pi^k \in [0,1]$: probability of facing attacker $k \in K$.

$D^k(j|c)$: utility of the defender when facing attacker $k \in K$, who attacks target $j \in J$ which is protected (covered).

$D^k(j|u)$: utility of the defender when facing attacker $k \in K$, who attacks target $j \in J$ which is unprotected (uncovered).

$A^k(j|c)$: utility of the attacker $k \in K$, attacking target $j \in J$ which is covered.

$A^k(j|u)$: utility of the attacker $k \in K$, attacking target $j \in J$ which is uncovered.

About utilities, it is assumed that for each $j \in J$ and $k \in K$, $D^k(j|c) \geq D^k(j|u)$, because defender wants to cover the attacked targets, and $A^k(j|u) \geq A^k(j|c)$, because attacker wants to attack uncovered targets.

Variables:

$x_{\mathrm{i}} \in [0,1]$: probability for defender to play pure strategy $\mathrm{i} \in I$, such that $\sum_{\mathrm{i}\in I} x_{\mathrm{i}} = 1$.

$c_j \in [0,1]$: probability of coverage of target $j \in J$, such that $c_j = \sum_{\mathrm{i}\in I: j\in J} x_{\mathrm{i}}, \forall j \in J$, i.e., is represented by the sum of the probabilities of all strategies that cover that target.

$q_j^k \in \{0,1\}$: indicates whether the attacker $k \in K$ strikes target $j \in J$.

The expected utility for the defender is:

$$\sum_{j\in J} \sum_{k\in K} \pi^k q_j^k \big\{ c_j D^k(j|c) + (1-c_j) D^k(j|u) \big\} \tag{3.4}$$

The expected utility for the k attacker is:

$$\sum_{j\in J} q_j^k \big\{ c_j A^k(j|c) + (1-c_j) A^k(j|u) \big\} \tag{3.5}$$

Figure (3.1) shows an example of security Stackelberg game. The map covers the Chilean northern border with Perú and Bolivia. Green and red location icons represent protected and unprotected targets (illegal immigration zones), respectively. Blue arrow is an attacker $k \in K$ who chooses a covered target $j \in J$, and gets the negative utility of $A^k(j|c)$, and the defender gets the positive utility of $D^k(j|c)$. Yellow arrow is an attacker $k \in K$ who chooses an uncovered target $j \in J$, and gets a positive utility of $A^k(j|u)$, and the defender gets the negative utility of $D^k(j|u)$. Second yellow line that crosses the location icon means that the attacker gets in the territory (and that is his utility or reward).

Figure 3.1: Graphic example of security Stackelberg game in the Chilean border.

The following model, (BLP-S$_{x,c,q}$), is a bilevel programming formulation for the Bayesian security Stackelberg game problem:

(BLP-S$_{x,c,q}$)

$$
\begin{aligned}
\max_{x,\,c,\,q} \quad & \sum_{j\in J}\sum_{k\in K} \pi^k q_j^k \big\{ c_j D^k(j|c) + (1-c_j) D^k(j|u) \big\} \\
\text{s.t} \quad & \sum_{\mathrm{i}\in I} x_{\mathrm{i}} = 1, \\
& x_{\mathrm{i}} \geq 0 \qquad \forall \mathrm{i} \in I, \\
& c_j = \sum_{\mathrm{i}\in I: j\in J} x_{\mathrm{i}} \qquad \forall j \in J, \\
& q^k \in \arg\max_{r^k} \left\{ \sum_{j\in J} r_j^k \big\{ c_j A^k(j|c) + (1-c_j) A^k(j|u) \big\} \right\} \qquad \forall k \in K, \\
& \quad r_j^k \in [0,1] \qquad \forall j \in J, \forall k \in K, \\
& \quad \sum_{j\in J} r_j^k = 1 \qquad \forall k \in K.
\end{aligned}
\tag{3.6}
$$

The objective function maximizes defender's expected utility. The first to third set of constraints characterize the exponentially many mixed strategies considered by the defender and relate them to coverage frequencies over the targets. Nested problem maximizes attacker's utility by attacking the target which represents his best response to defender's strategy.

A more compact formulation involving a polynomial number of variables and constraints can be obtained if projecting out the exponentially many x variables does not lead to exponentially many constraints. To show that the number of constraints in the projection is polynomial, [19] provides a mathematical proof that makes use of Farkas' Lemma [27]. This projection of x variables from constraints in the defender problem leads to the following constraints:

$$
\sum_{j\in J} c_j \leq m, \tag{3.7}
$$

$$
c \in [0,1]^{|J|}. \tag{3.8}
$$

Constraint (3.7) enforces that the total coverage provided to the targets cannot exceed m, the number of available security resources, and constraint (3.8) guarantees that the coverage probabilities are in range of probabilities.

This lead us to the following problem depending only on variables c and q.

(BLP-S$_{c,q}$)

$$
\begin{aligned}
\max_{c,\,q} \quad & \sum_{j\in J}\sum_{k\in K} \pi^k q_j^k \big\{ c_j D^k(j|c) + (1-c_j) D^k(j|u) \big\} \\
\text{s.t} \quad & \sum_{j\in J} c_j \le m, \\
& c \in [0,1]^{|J|}, \\
& q^k \in \arg\max_{r^k} \left\{ \sum_{j\in J} r_j^k \big\{ c_j A^k(j|c) + (1-c_j) A^k(j|u) \big\} \right\} \qquad \forall k \in K, \\
& \quad r_j^k \in [0,1] \qquad \forall j \in J, \forall k \in K, \\
& \quad \sum_{j\in J} r_j^k = 1 \qquad \forall k \in K.
\end{aligned}
\tag{3.9}
$$

Given an optimal solution to this formulation, a probability vector x can be obtained by solving the system of linear inequalities:

$$
\sum_{\mathrm{i}\in I} x_{\mathrm{i}} = 1, \qquad x_{\mathrm{i}} \ge 0 \qquad \forall \mathrm{i} \in I, \qquad c_j = \sum_{\mathrm{i}\in I : j\in J} x_{\mathrm{i}} \qquad \forall j \in J, \tag{3.10}
$$

This system involves $n+1$ equalities ($|J|+1$), there exist a solution in which the number of variables x_{i} with a strictly positive value is not larger than $n+1$, i.e., the output size of a SSG is polynomial in the input size.

In the next subsection, we present an algorithm that, given an optimal coverage probabilities on the targets, recovers an optimal mixed strategy that satisfies that coverage probabilities. This algorithm is used in [19].

3.2.1 The box method

The box method algorithm presented in this subsection provides a simple graphic way of recovering an optimal mixed strategy x that solves SSG formulation (BLP-S$_{x,c,q}$), and which is easily implementable by the leader, based on the vector c of optimal coverage probabilities returned by the compact SSG formulation (BLP-S$_{c,q}$).

The algorithm used for scheduling comes from [42]. In this paper, the authors need to schedule $J = 1, \ldots, n$ tasks on m processors. Tasks can be split in any numbers of ways in the processors, but processors cannot work on the same task at the same time. Each task $j \in J$ has a processing time c_j.

In our setting, we have m security resources, and we have n targets that need to be covered, and the processing time of a task corresponds to the coverage probability on target. We do not want to assign the same target twice in a single pure strategy i, because we want to cover most possible targets.

The following theorem is a particular case adapted from the general case presented in [42].

Theorem 3.1 If $\sum_{j \in J} c_j \leq m$, then a necessary and sufficient condition that there exists a distribution in which the coverage probabilities of all targets are considered is that $c_j \leq 1, \forall j \in J$.

Proof. The necessary condition is directly seen from $\sum_{j \in J} c_j \leq m$, because if it is not true, it is impossible to consider the coverage probabilities of all targets.

For the sufficient condition, assume each $c_j \leq 1$. We now show how the coverage probabilities can be considered so that after each coverage probability has been assigned, the following condition is fulfilled: *the total probability of every security resource (1) with possible exception of one, is either entirely filled up or is entirely empty.* Assigning target $j = 1$ entirely in the first security resource, there will be $1 - c_1$ probability left on the first security resource to be filled. The condition is fulfilled because we have just one security resource partially filled, or none in extreme case $c_1 = 1$.

Now assume the $j - 1$ targets, $1, 2, \ldots, j - 1$, have been assigned and the the condition is fulfilled. We show that it is possible to assign the j^{th} such that the condition is still fulfilled. Note that if $j \leq n$, after the first $j - 1$ targets have been assigned the total resources unassigned probabilities is greater than or equal to c_j, because $m \geq \sum_{j=1}^{n} c_j$ implies that $c_j \leq m - \sum_{k=1}^{j-1} c_k - \sum_{k=j+1}^{n} c_k$ and unassigned probability is $up = m - \sum_{k=j}^{n} c_k$. Which leads to $c_j \leq up$. But we don't want to assign the target j twice in the same pure strategy, so we cannot be sure that j can be assigned.

To handle this, we have three cases:

1. There is no security resource whose probability is partially filled: then assign j on an empty resource. The condition is still fulfilled.

   For the following cases, let the security resource R be the resource whose probability is not yet filled; suppose it has π units of probabilities left unassigned, and hence $1 - \pi$ assigned.

2. $\pi \geq c_j$: then assign j wholly on the security resource with the partially filled probability. The condition is still fulfilled.
3. $\pi < c_j$: assign j from $1 - \pi$ to 1, and from 0 to $c_j - \pi$ on a new resource. Since each $c_j \leq 1$, this guaranties that no target is assigned twice in a same pure strategy. The condition is still fullfilled.

Since these considerations are valid for $j \leq n$, the proof is complete.

□

The schedule is constructed as follows. One pile up the optimal c_j values consecutively inside m columns, which represent the security resources, of height one, from left to right. Whenever a column is filled, one can start filling up the next column with the remaining quantity of unassigned c_j from the previous column. After that, upon inspection of the resulting diagram, one can determine, before a processor switches task, all tasks that are

currently being processed and the time taken before a given configuration of tasks being processed changes.

In our setting these configuration correspond to the pure strategies (coverage of m targets) and the time taken between configuration changes , corresponds to the probability with which coverage configuration prior to the change occurs.

Algorithm 1 summarizes the steps required to recover the defender's implementable mixed strategy.

Algorithm 1: Box method

Input: c, m columns of height 1

Result: Defender's mixed strategy

1 Fill up optimal c_j inside the columns;

2 Make transversal cuts in the box every time there is a configuration change along the columns and final cut at height 1;

3 Read off the mixed strategy as follows:

1. Starting from the bottom of the box, the pure strategies correspond to the different configurations of resources covering target separated by the transversal cuts.
2. Each configuration's probability is given by the height of the corresponding configuration.

The following example shows the implementation of the box method, used to recover an optimal mixed strategy x, this was taken from [19].

Consider a security instance with $J = 4$, $K = 1$ and $m = 3$. Let the optimal coverage probabilities obtained by solving (3.9) be $c = (0.7, 0.7, 0.65, 0.95)$. Stacking up the values of the coverage probabilities c_j into $m = 3$ columns of the algorithm 1 produces the drawing of figure (3.2).

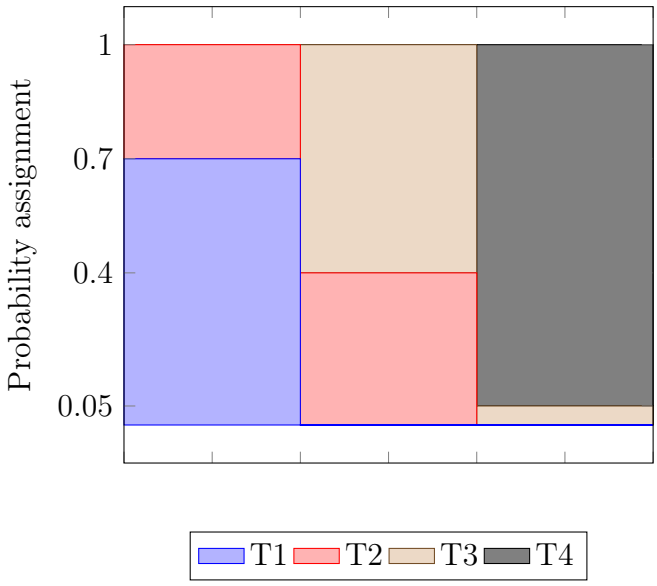

Figure 3.2: The box method.

As can be seen, we can do four transversal cuts in here. First cut in 0.05, second in 0.4, third in 0.7 and fourth in 1. The results are described in the next table:

Defender pure strategy	Targets protected	Weight in mixed strategy
1	(1,2,3)	0.05
2	(1,2,4)	0.35
3	(1,3,4)	0.3
4	(2,3,4)	0.3

Table 3.1: Optimal defender mixed strategy.

This show that in the 5% of times, defender will apply strategy 1, i.e, protect targets 1, 2 and 3. Strategy 2 will be applied 35% of times, protecting targets 1, 2 and 4. Defender will apply strategy 3 and 4 30% of times, protecting targets 1, 2 and 4; and 2, 3 and 4, respectively.

Chapter 4

General and security Stackelberg game formulations

4.1 Introduction

4.2 General Stackelberg games

4.2.1 Single level formulations

[46] tackle the problem of solving the bilevel formulation presented earlier, (BLP-G$_{x,q}$) in (3.3) by using a MILP reformulation based in optimality conditions [16] applied to the nested problem.

Given the second level convex problem linear relaxation for the k follower:

$$\begin{aligned} \max_{q} \quad & \sum_{\mathrm{i}\in I}\sum_{j\in J} C^k_{\mathrm{i}j} x_{\mathrm{i}} q^k_j \\ & q^k_j \geq 0 \qquad \forall j \in J, \\ & \sum_{j\in J} q^k_j = 1. \end{aligned} \tag{4.1}$$

Its dual is given by:

$$\begin{aligned} \min_{\nu} \quad & \nu^k \\ & \nu^k \geq \sum_{\mathrm{i}\in I} C^k_{\mathrm{i}j} x_{\mathrm{i}} \qquad \forall j \in J. \end{aligned} \tag{4.2}$$

Problem (4.1) can be replaced by the following set of constraints:

$$\sum_{j \in J} q_j^k = 1, \tag{4.3}$$

$$q_j^k \in \{0,1\} \qquad \forall j \in J, \tag{4.4}$$

$$0 \leq (s^k - \sum_{\mathrm{i} \in I} C_{\mathrm{ij}}^k x_{\mathrm{i}}) \leq (1 - q_j^k) M^2 \qquad \forall j \in J. \tag{4.5}$$

where $s^k \in \mathbb{R} \quad \forall k \in K$ and M^2 is a big-M constant. The two inequalities in constraint (4.5) ensure that $q_j^k = 1$ only for a pure strategy that maximizes the follower's payoff.

Mixed integer quadratic formulation

This transformation leads to the following formulation, (MIQP-G$_{x,q,s}$), which is a quadratic single level problem for the GSG where the only quadratic term appears in the objective function.

(MIQP-G$_{x,q,s}$)

$$\begin{aligned}
\max_{x,\,q,\,s} \quad & \sum_{\mathrm{i} \in I} \sum_{j \in J} \sum_{k \in K} \pi^k R_{\mathrm{ij}}^k x_{\mathrm{i}} q_j^k \\
\text{s.t} \quad & \sum_{\mathrm{i} \in I} x_{\mathrm{i}} = 1, \\
& x_{\mathrm{i}} \geq 0 \qquad \forall \mathrm{i} \in I, \\
& \sum_{j \in J} q_j^k = 1 \qquad \forall k \in K, \\
& q_j^k \in \{0,1\} \qquad \forall j \in J, \forall k \in K, \\
& 0 \leq (s^k - \sum_{\mathrm{i} \in I} C_{\mathrm{ij}}^k x_{\mathrm{i}}) \leq (1 - q_j^k) M^2 \qquad \forall j \in J, \forall k \in K, \\
& s \in \mathbb{R}^{|K|}.
\end{aligned} \tag{4.6}$$

Mixed integer linear formulations

The above formulation, (MIQP-G$_{x,q,s}$), can be linearized in two alternative ways.

In the first alternative, [46] eliminate non-linearity of the objective function in (MIQP-G$_{x,q,s}$) by adding an additional variable z that represents the product between x and q, $z_{\mathrm{ij}}^k = x_{\mathrm{i}} q_j^k \quad \forall \mathrm{i} \in I, \forall j \in J, \forall k \in K$. This action leads to the following formulation called (DOBSS$_{q,z,s}$)

.

(DOBSS$_{q,z,s}$)

$$
\begin{aligned}
\max_{q,\,z,\,s} \quad & \sum_{\mathrm{i}\in I}\sum_{j\in J}\sum_{k\in K}\pi^k R^k_{\mathrm{ij}} z^k_{\mathrm{ij}} \\
\text{s.t} \quad & \sum_{j\in J} q^k_j = 1 \qquad \forall k \in K, \\
& q^k_j \in \{0,1\} \qquad \forall j \in J, \forall k \in K, \\
& \sum_{j\in J} z^k_{\mathrm{ij}} = \sum_{j\in J} z^1_{\mathrm{ij}} \qquad \forall \mathrm{i} \in I, \forall k \in K, \\
& \sum_{\mathrm{i}\in I} z^k_{\mathrm{ij}} = q^k_j \qquad \forall j \in J, \forall k \in K, \\
& z^k_{\mathrm{ij}} \geq 0 \qquad \forall \mathrm{i} \in I, \forall j \in J, \forall k \in K, \\
& 0 \leq (s^k - \sum_{\mathrm{i}\in I}\sum_{h\in J} C^k_{\mathrm{ij}} z^k_{\mathrm{i}h}) \leq (1-q^k_j)M^2 \qquad \forall j \in J, \forall k \in K, \\
& s \in \mathbb{R}^{|K|}.
\end{aligned}
\tag{4.7}
$$

Second alternative to eliminate non-linearity, is presented by [36], and consists in avoid the quadratic term in the objective function by adding $|K|$ new variables and introducing a second family of constraints involving a big-M constant, this is called ($\mathrm{D2}_{x,q,s,f}$).

($\mathrm{D2}_{x,q,s,f}$)

$$
\begin{aligned}
\max_{x,\,q,\,s,\,f} \quad & \sum_{k\in K}\pi^k f^k \\
\text{s.t} \quad & \sum_{\mathrm{i}\in I} x_{\mathrm{i}} = 1, \\
& x_{\mathrm{i}} \geq 0 \qquad \forall \mathrm{i} \in I, \\
& \sum_{j\in J} q^k_j = 1 \qquad \forall k \in K, \\
& q^k_j \in \{0,1\} \qquad \forall j \in J, \forall k \in K, \\
& 0 \leq (s^k - \sum_{\mathrm{i}\in I} C^k_{\mathrm{ij}} x_{\mathrm{i}}) \leq (1-q^k_j)M^2 \qquad \forall j \in J, \forall k \in K, \\
& f^k \leq \sum_{\mathrm{i}\in I} R^k_{\mathrm{ij}} x_{\mathrm{i}} + (1-q^k_j)M^1 \qquad \forall j \in J, \forall k \in K, \\
& s, f \in \mathbb{R}^{|K|}.
\end{aligned}
\tag{4.8}
$$

Projected formulations

Additionally, we can get two new equivalent formulations by applying Fourier-Motzkin elimination [25] to ($\mathrm{DOBSS}_{q,z,s}$) and ($\mathrm{D2}_{x,q,s,f}$) respectively. This procedure is applied in [19].

Fourier-Motzkin elimination (FME) is an algorithm useful to eliminate variables of a linear inequalities system. It creates a new system without the selected variable, but with the same

solution in the remaining variables. We apply this algorithm to eliminate s^k variables, we are not interested in their solutions, because are dual variables of the nested problem of bilevel formulation of the Bayesian GSG, (BLP-G$_{x,q}$).

First, we apply FME to (DOBSS$_{q,z,s}$). This leads the constraint:

$$0 \leq (s^k - \sum_{\mathrm{i} \in I} \sum_{h \in J} C^k_{\mathrm{i}j} z^k_{\mathrm{i}h}) \leq (1 - q^k_j) M^2 \qquad \forall j \in J, \forall k \in K, \tag{4.9}$$

to the new constraint:

$$\sum_{\mathrm{i} \in I} \sum_{h \in J} (C^k_{\mathrm{i}j} - C^k_{\mathrm{i}l}) z^k_{\mathrm{i}h} \leq (1 - q^k_l) M^2 \qquad \forall j, l \in J, \forall k \in K. \tag{4.10}$$

This elimination yields (DOBSS$_{q,z}$).

Second, we apply FME to (D2$_{x,q,s,f}$). This changes the constraint:

$$0 \leq (s^k - \sum_{\mathrm{i} \in I} C^k_{\mathrm{i}j} x_{\mathrm{i}}) \leq (1 - q^k_j) M^2 \qquad \forall j \in J, \forall k \in K, \tag{4.11}$$

for the new constraint:

$$\sum_{\mathrm{i} \in I} (C^k_{\mathrm{i}j} - C^k_{\mathrm{i}l}) x_{\mathrm{i}} \leq (1 - q^k_l) M^2 \qquad \forall j, l \in J, \forall k \in K. \tag{4.12}$$

This yields (D2$_{x,q,f}$).

[19] analyzes the behavior of (DOBSS$_{q,z}$) and (D2$_{x,q,f}$) compared to their unprojected counterparts to see if removing variables s^k at the expense of adding constraints is worthwhile.

Reformulation-linearization technique formulation

Reformulation-linearization technique (RLT) is a technique mainly used to generate tight equivalent representations for mixed integer (binary) programming problems. It consists of multiplying a constraint by a variable, and after linearizing it by change of variable. This technique is described in [51].

We now present a new formulation, referred as (MIP-p-G$_{q,z}$) by [19]. This formulation is mainly composed by (DOBSS$_{q,z,s}$) mixed with RLT projected constraints (4.12) of (D2$_{x,q,f}$), and is a Bayesian extension to a single follower formulation in [22].

It consists in replacing constraints:

$$0 \leq (s^k - \sum_{\mathrm{i} \in I} \sum_{h \in J} C^k_{\mathrm{i}j} z^k_{\mathrm{i}h}) \leq (1 - q^k_j) M^2 \qquad \forall j \in J, \forall k \in K, \tag{4.13}$$

for the constraints

$$\sum_{\mathrm{i} \in I} (C^k_{\mathrm{i}j} - C^k_{\mathrm{i}l}) z^k_{\mathrm{i}j} \geq 0 \qquad \forall j, l \in J, \forall k \in K. \tag{4.14}$$

This yields (MIP-p-G$_{q,z}$):

$$\begin{aligned}
\max_{q,\, z} \quad & \sum_{\mathrm{i} \in I} \sum_{j \in J} \sum_{k \in K} \pi^k R^k_{\mathrm{i}j} z^k_{\mathrm{i}j} \\
\text{s.t} \quad & \sum_{j \in J} q^k_j = 1 \qquad \forall k \in K, \\
& q^k_j \in \{0,1\} \qquad \forall j \in J, \forall k \in K, \\
& \sum_{j \in J} z^k_{\mathrm{i}j} = \sum_{j \in J} z^1_{\mathrm{i}j} \qquad \forall \mathrm{i} \in I, \forall k \in K, \\
& \sum_{\mathrm{i} \in I} z^k_{\mathrm{i}j} = q^k_j \qquad \forall j \in J, \forall k \in K, \\
& z^k_{\mathrm{i}j} \geq 0 \qquad \forall \mathrm{i} \in I, \forall j \in J, \forall k \in K, \\
& \sum_{\mathrm{i} \in I} (C^k_{\mathrm{i}j} - C^k_{\mathrm{i}l}) z^k_{\mathrm{i}j} \geq 0 \qquad \forall j, l \in J, \forall k \in K.
\end{aligned} \tag{4.15}$$

Briefly, constraints (4.14) were derived by multiplying projected constraints (4.12) by q^k_l. It makes the right hand side to be 0, because it is a binary variable. About the left hand side, it has a nonlinear term $x_{\mathrm{i}} q^k_l$ and can be changed by $x_{\mathrm{i}} q^k_j$ and change the inequality sense. Finally, this term is linearized using change of variables $x_{\mathrm{i}} q^k_j = z^k_{\mathrm{i}j}$ as in (DOBSS$_{q,z,s}$).

4.2.2 Theoretical comparison of the formulations

[46] proves the equivalence of (DOBSS$_{q,z,s}$) and (MIQP-G$_{x,q,s}$). On the other hand, [36] gives the proof that (D2$_{x,q,s,f}$) is equivalent to (MIQP-G$_{x,q,s}$). Therefore, (DOBSS$_{q,z,s}$) and (D2$_{x,q,s,f}$) are equivalent formulations.

[19] proves that (MIP-p-G$_{q,z}$) is equivalent to (MIQP-G$_{x,q,s}$).

Finally, by FME [25], (DOBBS$_{q,z}$) and (D2$_{x,q,f}$), the projected formulations, are both equivalent to their respective unprojected formulations (DOBSS$_{q,z,s}$) and (D2$_{x,q,s,f}$).

Hence, formulations so far are all equivalent.

Linear relaxations

In this section, the formulations are compared about their linear relaxations. The results shown here are obtained by [19].

Given a formulation F, we define the following concepts:

$\bar{F}$: linear relaxation of F.

$\mathcal{P}(\bar{F})$: feasible region of $\bar{F}$.

$v(F)$: optimal value of F.

[19] proofs that:

$$\mathcal{P}(\overline{MIP-p-G_{q,z}}) \subseteq \mathcal{P}(\overline{DOBSS_{q,z}}) \subseteq \mathcal{P}(\overline{D2_{x,q,s,f}}), \tag{4.16}$$

in consequence:

$$v(\overline{MIP-p-G_{q,z}}) \leq v(\overline{DOBSS_{q,z}}) \leq v(\overline{D2_{x,q,s,f}}). \tag{4.17}$$

.

4.2.3 Computational comparison of the formulations

[19] realized computational experiments for the formulations presented in the previous section. We sumarize his findings in this section.

He solved instances where the payoff matrices for the leader and follower were randomly generated in two ways:

1. 100% of elements randomly generated between 0 and 10 (referred to as instances with no variability).
2. 90% of elements randomly generated between 0 and 10 and 10% of elements between 0 and 100 (referred to as instances with variability).

About instances size, he considered instances for $|I| \in \{10, 20, 30\}$, $|J| \in \{10, 20, 30\}$ and $|K| \in \{2, 4, 6\}$. Generating, for each size, 5 instances of no variability, and 5 instances with variability. In total, 135 instances of each variability level.

He compared formulations in 4 metrics: total running time to solve the integer problem, running time to solve the linear relaxation of the integer problem, total number of nodes explored in the branch & bound solving scheme and gap percentage in the root node. We summarize his results:

Total running time to solve the integer problem

In this point of comparison, when there is no variability (DOBSS$_{q,z,s}$) and (MIP-p-G$_{q,z}$) are the two most competitive formulations. (D2$_{x,q,s,f}$) can be solved efficiently for instances not too big, but not in difficult instances.

When variability in the payoff matrices is introduced, best performance comes from (MIP-p-G$_{q,z}$) and (DOBSS$_{q,z,s}$) again, followed by (D2$_{x,q,s,f}$) which comes noncompetitive when instances are difficult.

Running time to solve the linear relaxation of the integer problem

According to linear relaxation, when there is no variability, the best performance formulation is (D2$_{x,q,s,f}$) and the worst performance comes from (MIP-p-G$_{q,z}$).

When variability is introduced, best performance formulation is the same, and worst changes to (DOBSS$_{q,z}$), but (MIP-p-G$_{q,z}$) is the second worst formulation.

Total number of nodes explored

(MIP-p-G$_{q,z}$) is the formulation that uses the fewest nodes, in both variability levels. This makes sense, because is the tightest formulation. On the other hand, formulation that uses the biggest nodes in both variability levels is (D2$_{x,q,s,f}$).

Gap percentage in the root node

According to the gap obtained across the instances solved, (MIP-p-G$_{q,z}$) is dramatically the best formulation, (DOBSS$_{q,z,s}$) is in the middle and (D2$_{x,q,s,f}$) is the worst.

According to the results described, we choose two formulations to make an hybrid algorithm, a fast linearly relaxed formulation, and a tight formulation. Fast formulation chosen, according to the theoretical and computational proofs, is (D2$_{x,q,s,f}$). The tight formulation chosen is (MIP-p-G$_{q,z}$), which is transformed by changing the constraint

$$\sum_{j\in J} z^k_{\mathrm{i}j} = \sum_{j\in J} z^1_{\mathrm{i}j} \qquad \forall \mathrm{i} \in I, \forall k \in K, \tag{4.18}$$

for the constraint

$$\sum_{j\in J} z^k_{\mathrm{i}j} = x_{\mathrm{i}} \qquad \forall \mathrm{i} \in I, \forall k \in K. \tag{4.19}$$

This yiels (MIP-p-G$_{x,q,z}$), that will be shown in the next chapter.

4.3 Security Stackelberg games

4.3.1 Single level formulations

In this section, we derive single level formulations applying to security the same procedure of [46] in the general setting, to compact security formulation (BLP-$S_{c,q}$) in (3.9), i.e., we apply optimality conditions to the nested security problem.

Given the second level convex problem linear relaxation for the k attacker:

$$\begin{aligned} \max_{q} \quad & \sum_{j\in J} q_j^k \big\{ c_j A^k(j|c) + (1-c_j) A^k(j|u) \big\} \\ & q_j^k \geq 0 \qquad \forall j \in J, \\ & \sum_{j\in J} q_j^k = 1. \end{aligned} \tag{4.20}$$

Its dual is given by:

$$\begin{aligned} \min_{\nu} \quad & \nu^k \\ & \nu^k \geq c_j A^k(j|c) + (1-c_j) A^k(j|u) \qquad \forall j \in J. \end{aligned} \tag{4.21}$$

(4.20) can be replaced by the following set of constraints:

$$\sum_{j\in J} q_j^k = 1, \tag{4.22}$$

$$q_j^k \in \{0,1\} \qquad \forall j \in J, \tag{4.23}$$

$$0 \leq (s^k - c_j A^k(j|c) - (1-c_j) A^k(j|u) \leq (1-q_j^k) M^2 \qquad \forall j \in J. \tag{4.24}$$

where $s^k \in \mathbb{R}\ \forall k \in K$ and M^2 is a big-M constant. The two constraints (4.24) ensure that $q_j^k = 1$ only for a pure strategy that maximizes the attacker's payoff.

Mixed integer quadratic formulation

This transformation leads us to the following formulation (MIQP-$G_{c,q,s}$) which is a quadratic single level problem for the SSG, where the only quadratic term appears in the objective function.

(MIQP-$S_{c,q,s}$)

$$
\begin{aligned}
\max_{c,\,q,\,s} \quad & \sum_{j\in J}\sum_{k\in K} \pi^k q_j^k \left\{ c_j D^k(j|c) + (1-c_j) D^k(j|u) \right\} \\
\text{s.t} \quad & \sum_{j\in J} c_j \leq m, \\
& c \in [0,1]^{|J|}, \\
& \sum_{j\in J} q_j^k = 1 \qquad \forall k \in K, \\
& q_j^k \in \{0,1\} \qquad \forall j \in J, \forall k \in K, \\
& 0 \leq (s^k - c_j A^k(j|c) - (1-c_j) A^k(j|u) \leq (1-q_j^k) M^2 \qquad \forall j \in J, \forall k \in K, \\
& s \in \mathbb{R}^{|K|}.
\end{aligned}
\tag{4.25}
$$

Mixed integer linear formulations

The above formulation, (MIQP-$S_{c,q,s}$), can be linearized in two alternative ways.

In the first alternative, [46] eliminate non-linearity of the objective function in (MIQP-$S_{c,q,s}$) by adding an additional variable y that represents the product between C and q, $y_{jj}^k = c_j q_j^k \quad \forall j \in J, \forall k \in K$. This action leads to the following formulation called SDOBSS$_{q,s,y}$ in [19].

(SDOBSS$_{q,s,y}$)

$$
\begin{aligned}
\max_{q,\,s,\,y} \quad & \sum_{j\in J}\sum_{k\in K} \left\{ \pi^k (D^k(j|c) y_{jj}^k + D^k(j|u)(q_j^k - y_{jj}^k)) \right\} \\
\text{s.t} \quad & \sum_{j\in J} q_j^k = 1 \qquad \forall k \in K, \\
& q_j^k \in \{0,1\} \qquad \forall j \in J, \forall k \in K, \\
& \sum_{j\in J} y_{lj}^k = \sum_{j\in J} y_{lj}^1 \qquad \forall l \in J, \forall k \in K, \\
& \sum_{l\in J} y_{lj}^k \leq m q_j^k \qquad \forall j \in J, \forall k \in K, \\
& 0 \leq y_{lj}^k \leq q_j^k \qquad \forall l, j \in J, \forall k \in K, \\
& 0 \leq s^k - A^k(j|c) \sum_{h\in J} y_{jh}^k - A^k(j|u)(1 - \sum_{h\in J} y_{jh}^k) \leq \\
& (1-q_j^k) M^2 \qquad \forall j \in J, \forall k \in K, \\
& s \in \mathbb{R}^{|K|}.
\end{aligned}
\tag{4.26}
$$

Second alternative to eliminate non-linearity, is presented by [36], and consists in avoiding the quadratic term in the objective function by adding $|K|$ new variables and introducing a second family of constraints involving a big-M constant, this is called (ERASER$_{c,q,s,f}$).

(ERASER$_{c,q,s,f}$)

$$
\begin{aligned}
\max_{c,\,q,\,s,\,f} \quad & \sum_{k\in K} \pi^k f^k \\
\text{s.t} \quad & \sum_{j\in J} c_j \leq m, \\
& 0 \leq c_j \leq 1 \quad \forall j \in J, \\
& \sum_{j\in J} q_j^k = 1 \quad \forall k \in K, \\
& q_j^k \in \{0,1\} \quad \forall j \in J, \forall k \in K, \\
& f^k \leq D^k(j|c)c_j + D^k(j|u)(1-c_j) \\
& \qquad + (1-q_j^k)M^1 \quad \forall j \in J, \forall k \in K, \\
& 0 \leq s^k - A^k(j|c)c_j - A^k(j|u)(1-c_j) \leq \\
& (1-q_j^k)M^2 \quad \forall j \in J, \forall k \in K, \\
& s, f \in \mathbb{R}^K.
\end{aligned}
\tag{4.27}
$$

Projected formulations

As in GSG, we can get two new equivalent formulations projecting out the s variables by applying Fourier-Motzkin elimination [25] to (SDOBSS$_{q,s,y}$) and (ERASER$_{c,q,s,f}$) respectively. This procedure is applied in [19].

First, we apply FME to (SDOBSS$_{q,s,y}$). This leads the constraint:

$$
0 \leq s^k - A^k(j|c)\sum_{h\in J} y_{jh}^k - A^k(j|u)(1-\sum_{h\in J} y_{jh}^k) \leq (1-q_j^k)M^2 \qquad \forall j \in J, \forall k \in K, \tag{4.28}
$$

to the new constraint:

$$
\begin{aligned}
(A^k(j|c) - A^k(j|u))\sum_{g\in J} y_{jg}^k + (A^k(l|u) - A^k(l|c))\sum_{g\in J} y_{lg}^k + \\
A^k(j|u) - A^k(l|u) \leq (1-q_l^k)M^2 \qquad j,l \in J, \forall k \in K.
\end{aligned}
\tag{4.29}
$$

This constraint transformation yields (SDOBSS$_{q,y}$).

Second, we apply FME to (ERASER$_{c,q,s,f}$). This changes the constraint:

$$
\begin{aligned}
0 \leq s^k - A^k(j|c)c_j - A^k(j|u)(1-c_j) \leq \\
(1-q_j^k)M^2 \quad \forall j \in J, \forall k \in K,
\end{aligned}
\tag{4.30}
$$

for the new constraint:

$$\begin{aligned}(A^k(j|c) - A^k(j|u))c_j + (A^k(l|u) - A^k(l|c))c_l + \\ A^k(j|u) - A^k(l|u) \leq (1 - q_l^k)M^2 \qquad \forall j, l \in J, \forall k \in K.\end{aligned} \tag{4.31}$$

This yields (ERASER$_{c,q,f}$).

Reformulation-linearization technique formulation

We now present a new formulation, referred as (MIP-p-S$_{q,y}$) by [19]. This formulation is mainly composed by (SDOBSS$_{q,s,y}$) mixed with RLT projected constraints (4.31) of (ERASER$_{c,q,f}$). This new formulation is a Bayesian extension to a single follower formulation in [22].

It consists in replacing constraints:

$$0 \leq s^k - A^k(j|c)\sum_{h \in J} y_{jh}^k - A^k(j|u)(1 - \sum_{h \in J} y_{jh}^k) \leq (1 - q_j^k)M^2 \qquad \forall j \in J, \forall k \in K, \tag{4.32}$$

for the constraints

$$\begin{aligned}A^k(j|c)y_{jj}^k + A^k(j|u)(q_j^k - y_{jj}^k) - A^k(l|c)y_{lj}^k - \\ A^k(l|u)(q_j^k - y_{lj}^k) \geq 0 \qquad \forall j, l \in J, \forall k \in K.\end{aligned} \tag{4.33}$$

This yields (MIP-p-S$_{q,y}$):

(MIP-p-S$_{q,y}$)

$$\begin{aligned}
\max_{q,\,y} \quad & \sum_{j \in J}\sum_{k \in K} \left\{\pi^k(D^k(j|c)y_{jj}^k + D^k(j|u)(q_j^k - y_{jj}^k))\right\} \\
\text{s.t} \quad & \sum_{j \in J} q_j^k = 1 \qquad \forall k \in K, \\
& q_j^k \in \{0,1\} \qquad \forall j \in J, \forall k \in K, \\
& \sum_{j \in J} y_{lj}^k = \sum_{j \in J} y_{lj}^1 \qquad \forall l \in J, \forall k \in K, \\
& \sum_{l \in J} y_{lj}^k \leq m q_j^k \qquad \forall j \in J, \forall k \in K, \\
& 0 \leq y_{lj}^k \leq q_j^k \qquad \forall l, j \in J, \forall k \in K, \\
& A^k(j|c)y_{jj}^k + A^k(j|u)(q_j^k - y_{jj}^k) - \\
& A^k(l|c)y_{lj}^k - A^k(l|u)(q_j^k - y_{lj}^k) \geq 0 \qquad \forall j, l \in J, \forall k \in K.
\end{aligned} \tag{4.34}$$

Briefly, constraints (4.33) were derived by multiplying projected constraints (4.31) by q_l^k. It makes the right hand side to be 0, because it is a binary variable. About the left hand side, it has a nonlinear term $c_j q_l^k$ and can be changed by $c_j q_j^k$ and change the inequality sense. Finally, this term is linearized using change of variables $c_j q_j^k = y_{jj}^k$ as in (SDOBSS$_{q,s,y}$).

4.3.2 Theoretical comparison of the formulations

[46] proves the equivalence of (SDOBSS$_{q,s,y}$) and (MIQP-S$_{c,q,s}$). On the other hand, [36] gives the proof that (ERASER$_{c,q,s,f}$) is equivalent to (MIQP-S$_{c,q,s}$). Therefore, (SDOBSS$_{q,s,y}$) and (ERASER$_{c,q,s,f}$) are equivalent formulations.

[19] proves that (MIP-p-S$_{q,y}$) is equivalent to (MIQP-S$_{c,q,s}$).

Finally, by FME [25], (SDOBBS$_{q,y}$) and (ERASER$_{c,q,f}$), the projected formulations, are both equivalent to their respective unprojected formulations (DOBSS$_{q,s,y}$) and (ERASER$_{c,q,s,f}$).

Hence, formulations so far are all equivalent.

Linear relaxations

In this section, the formulations are compared about their linear relaxations. The results shown here are obtained by [19].

Notation used in this section is defined in section (4.2.2).

[19] proofs that:

$$\mathcal{P}(\overline{MIP-p-S_{q,y}}) \subseteq \mathcal{P}(\overline{SDOBSS_{q,y}}) \subseteq \mathcal{P}(\overline{ERASER_{c,q,s,f}}), \tag{4.35}$$

in consequence:

$$v(\overline{MIP-p-S_{q,y}}) \leq v(\overline{SDOBSS_{q,y}}) \leq v(\overline{ERASER_{c,q,s,f}}). \tag{4.36}$$

4.3.3 Computational comparison of the formulations

As in GSG setting, [19] realized computational experiments for the formulations presented in the previous section. We sumarize his findings in this section.

He solved instances where the payoff matrices for the leader and follower were randomly generated in two ways:

1. 100% of elements of penalty matrices randomly generated between 0 and 5, and reward matrices between 5 and 10 (referred to as no variability instances).
2. 90% of elements of penalty matrices randomly generated between 0 and 5, and reward matrices between 5 and 10. 10% of elements of penalty matrices between 0 and 50, and reward matrices between 50 and 100 (referred to as variability instances).

About instances size, he considered different parameters for no variability and variability instances. Considering 5 instances of each size in each variability level, 300 instances in total.

For no variability instances, $|J| \in \{10, 20, 30, 40, 50\}$, $|K| \in \{2, 4, 6, 8\}$, $m = \{25\%, 50\%, 75\%\}$ of the number of targets.

For variability instances, $|J| \in \{30, 40, 50, 60, 70\}$, $|K| \in \{6, 8, 10, 12\}$, $m = \{25\%, 50\%, 75\%\}$ of the number of targets.

We summarize his results:

Total running time to solve the integer problem

When there is no variability (MIP-p-$S_{q,y}$) is the most competitive formulation for difficult instances to solve the integer problem, while (ERASER$_{c,q,s,f}$) is for easier instances. (SDOBBS$_{q,y,s}$) has the worst performance.

When payoff matrices has variability, (ERASER$_{c,q,s,f}$) is the best formulation in almost all instances, except for the very difficult ones.

Running time to solve the linear relaxation of the integer problem

According to linear relaxation, (ERASER$_{c,q,s,f}$) is dramatically better than the other formulations, in any variability level. (MIP-p-$S_{q,y}$) is the slowest. This makes sense, since (MIP-p-$S_{q,y}$) is the tightest formulation and (ERASER$_{c,q,s,f}$) is the lightest.

Total number of nodes explored

(MIP-p-$S_{q,y}$) is the formulation that uses the fewest nodes, in both variability levels. This makes sense, because is the tightest formulation. On the other hand, formulation that uses the biggest nodes in both variability levels is (ERASER$_{c,q,s,f}$).

Gap percentage in the root node

According to the gap obtained in the root node across the instances solved, (MIP-p-$S_{q,y}$) is dramatically the best formulation, (SDOBSS$_{q,y,s}$) is in the middle and (ERASER$_{c,q,s,f}$) is the worst, just as in GSG.

According to the results described, we choose two formulations to make an hybrid algorithm, a fast linearly relaxed formulation, and a tight formulation, just as in the general setting. Fast formulation chosen, according to the theoretical and computational proofs, is (ERASER$_{c,q,s,f}$). The tight formulation chosen is (MIP-p-$S_{q,y}$), which is transformed by changing the constraint

$$\sum_{j \in J} y_{lj}^{k} = \sum_{j \in J} y_{lj}^{1} \qquad \forall l \in J, \forall k \in K, \tag{4.37}$$

for the constraint

$$\sum_{j \in J} y_{lj}^{k} = c_l \qquad \forall l \in J, \forall k \in K. \tag{4.38}$$

This yiels (MIP-p-S$_{c,q,y}$), that will be shown in the next chapter.

Chapter 5

Benders decomposition approaches

5.1 Introduction

In this chapter, we derive the decomposition approaches described in §2.4.1, whose results will be used in next chapters. These decompositions are applied to the tight formulations chosen in the previous chapter, (MIP-p-G$_{x,q,z}$) and (MIP-p-S$_{c,q,y}$), for general and security Stackelberg games respectively.

5.2 General Stackelberg games

The approaches of this section are derived from formulation (MIP-p-G$_{x,q,z}$):

$$
\begin{aligned}
\max_{x,\,q,\,z} \quad & \sum_{\mathrm{i}\in I}\sum_{j\in J}\sum_{k\in K}\pi^k R^k_{\mathrm{i}j} z^k_{\mathrm{i}j} \\
\text{s.t} \quad & \sum_{j\in J} q^k_j = 1 \quad \forall k \in K, \\
& q^k_j \in \{0,1\} \qquad \forall j \in J, \forall k \in K, \\
& \sum_{j\in J} z^k_{\mathrm{i}j} = x_{\mathrm{i}} \qquad \forall \mathrm{i} \in I, \forall k \in K, \\
& \sum_{\mathrm{i}\in I} z^k_{\mathrm{i}j} = q^k_j \qquad \forall j \in J, \forall k \in K, \\
& z^k_{\mathrm{i}j} \geq 0 \qquad \forall \mathrm{i} \in I, \forall j \in J, \forall k \in K, \\
& \sum_{\mathrm{i}\in I} (C^k_{\mathrm{i}j} - C^k_{\mathrm{i}l}) z^k_{\mathrm{i}j} \geq 0 \qquad \forall l, j \in J, \forall k \in K.
\end{aligned}
\tag{5.1}
$$

We first apply standard Benders decomposition to this formulation, keeping q in master problem. Additionally, we derive a special normalized approach as described in section §2.4.2 with the same variables in master problem. We further derive normalized approaches for standard Benders decomposition approaches in [19], which includes x and x and q in master

problem.

5.2.1 q remains in master problem

The first Benders decomposition approach derived is when q remains in master problem, and x and z are sent to subproblem.

Recall that the subproblem contains the constraints where the variables x or z appear. Additionally, it contains the parts of the objective function where the same variables are, leaving in the master problem the parts containing exclusively q variables.

Subproblem

$$\begin{aligned}
\max_{x,\,z} \quad & \sum_{k\in K}\sum_{\mathrm{i}\in I}\sum_{j\in J}\pi^k R^k_{\mathrm{i}j} z^k_{\mathrm{i}j} \\
\text{s.t} \quad & \sum_{j\in J} z^k_{\mathrm{i}j} - x_{\mathrm{i}} = 0 \quad \forall \mathrm{i}\in I, \forall k\in K \quad (\alpha^k_{\mathrm{i}}), \\
& \sum_{\mathrm{i}\in I} z^k_{\mathrm{i}j} = q^k_j \quad \forall j\in J, \forall k\in K \quad (\beta^k_j), \\
& \sum_{\mathrm{i}\in I} (C^k_{\mathrm{i}j} - C^k_{\mathrm{i}l}) z^k_{\mathrm{i}j} \geq 0 \quad \forall l,j\in J, \forall k\in K \quad (\gamma^k_{jl}), \\
& z^k_{\mathrm{i}j} \geq 0 \quad \forall \mathrm{i}\in I, \forall j\in J, \forall k\in K.
\end{aligned} \tag{5.2}$$

The separation problem where the cuts are generated is the following:

Separation problem

This is the dual of subproblem:

$$\begin{aligned}
\min_{\alpha,\,\beta,\,\gamma} \quad & \sum_{j\in J}\sum_{k\in K} \beta^k_j q^k_j \\
\text{s.t} \quad & \sum_{k\in K} \alpha^k_{\mathrm{i}} = 0 \quad \forall \mathrm{i}\in I, \\
& \alpha^k_{\mathrm{i}} + \beta^k_j + \sum_{l\in J: l\neq j} (C^k_{\mathrm{i}j} - C^k_{\mathrm{i}l})\gamma^k_{jl} \geq \\
& \pi^k R^k_{\mathrm{i}j} \quad \forall \mathrm{i}\in I, \forall j\in J, \forall k\in K, \\
& \gamma^k_{jl} \leq 0 \quad \forall \mathrm{i}\in I, \forall j\in J, \forall k\in K.
\end{aligned} \tag{5.3}$$

Finally, these are the cuts to be added to master problem:

$$\theta \leq \sum_{j\in J}\sum_{k\in K} \beta^{kg^*}_j q^k_j \quad \forall g \in \text{extreme points}, \tag{5.4}$$

$$0 \leq \sum_{j \in J} \sum_{k \in K} \delta_j^{kh^*} q_j^k \quad \forall h \in \text{extreme rays.} \tag{5.5}$$

5.2.2 q remains in master problem, normalized

The second Benders decomposition approach is an extension of the previous, i.e., q appears in master problem and x and z in subproblem, but in addition we apply a normalization procedure described in [29] and [30].

From the previous approach, we know that the cut generation procedure is the following:

We first solve the separation problem. If the problem is unbounded, we generate the following feasibility cut:

$$0 \leq \sum_{j \in J} \sum_{k \in K} \delta_j^{k^*} q_j^k, \tag{5.6}$$

Otherwise, if we get an optimal solution such that

$$\sum_{j \in J} \sum_{k \in K} \beta_j^{k^*} q_j^{k^*} < \theta^*, \tag{5.7}$$

we generate the following optimality cut

$$\sum_{j \in J} \sum_{k \in K} \beta_j^{k^*} q_j^k \geq \theta. \tag{5.8}$$

So, we generate cuts if and only if the following polyhedron is infeasible:

Cut generation problem

$$\begin{aligned}
&\sum_{k \in K} \sum_{\mathrm{i} \in I} \sum_{j \in J} \pi^k R_{\mathrm{ij}}^k z_{\mathrm{ij}}^k \geq \theta^* \quad (\lambda), \\
&\sum_{j \in J} z_{\mathrm{ij}}^k - x_{\mathrm{i}} = 0 \quad \forall \mathrm{i} \in I, \forall k \in K \quad (\alpha_{\mathrm{i}}^k), \\
&\sum_{\mathrm{i} \in I} z_{\mathrm{ij}}^k = q_j^{k^*} \quad \forall j \in J, \forall k \in K \quad (\beta_j^k), \\
&\sum_{\mathrm{i} \in I} (C_{\mathrm{ij}}^k - C_{\mathrm{il}}^k) z_{\mathrm{ij}}^k \geq 0 \quad \forall l, j \in J, \forall k \in K \quad (\gamma_{jl}^k), \\
&\quad z_{\mathrm{ij}}^k \geq 0 \quad \forall \mathrm{i} \in I, \forall j \in J, \forall k \in K.
\end{aligned} \tag{5.9}$$

Which is the same than finding dual unboundedness:

Dual cut generation problem

$$
\begin{aligned}
\min_{\alpha,\beta,\gamma,\lambda} \quad & \sum_{j\in J}\sum_{k\in K}\beta_j^k q_j^{k^*} + \lambda\theta^* \\
\text{s.t} \quad & \sum_{k\in K}\alpha_{\mathrm{i}}^k = 0 \quad \forall \mathrm{i} \in I, \\
& \alpha_{\mathrm{i}}^k + \beta_j^k + \sum_{l\in J: l\neq j}(C_{\mathrm{i}j}^k - C_{\mathrm{i}l}^k)\gamma_{jl}^k + \\
& \qquad \lambda\pi^k R_{\mathrm{i}j}^k \geq 0 \quad \forall \mathrm{i} \in I, \forall j \in J, \forall k \in K, \\
& \gamma_{jl}^k \leq 0 \quad \forall \mathrm{i} \in I, \forall j \in J, \forall k \in K, \\
& \lambda \leq 0.
\end{aligned}
\tag{5.10}
$$

Hence, if this occurs, the cut to be generated is:

$$
0 \leq \sum_{j\in J}\sum_{k\in K}\delta_j^{kh^*} q_j^k + \lambda^*\theta \tag{5.11}
$$

We now apply the procedure described in [29] and [30].

Finding unboundedness in (5.10) consists of finding the set of extreme rays that satisfy:

$$
\sum_{j\in J}\sum_{k\in K}\beta_j^k q_j^k + \lambda\theta < 0, \tag{5.12}
$$

this because if the objective function of this problem is less than 0, this function goes to $-\infty$. But if we could fix the objective function to a negative value different to this, the problem would still be a unboundedness problem, but represented as an optimality problem. We now set the optimal value to:

$$
\sum_{j\in J}\sum_{k\in K}\beta_j^k q_j^k + \lambda\theta = -1, \tag{5.13}
$$

So now we can find extreme points that represent extreme rays.

With this, we get a polyhedron known in [31] as *alternative polyhedron*:

Alternative polyhedron

$$\begin{aligned}
&\sum_{k\in K} \alpha_{\mathrm{i}}^k = 0 \quad \forall \mathrm{i} \in I, \\
&\alpha_{\mathrm{i}}^k + \beta_j^k + \sum_{l\in J: l\neq j} (C_{\mathrm{i}j}^k - C_{\mathrm{i}l}^k)\gamma_{jl}^k + \\
&\qquad \lambda \pi^k R_{\mathrm{i}j}^k \geq 0 \quad \forall \mathrm{i} \in I, \forall j \in J, \forall k \in K, \\
&\gamma_{jl}^k \leq 0 \quad \forall \mathrm{i} \in I, \forall j \in J, \forall k \in K, \\
&\lambda \leq 0, \\
&\sum_{j\in J}\sum_{k\in K} \beta_j^k q_j^k + \lambda\theta = -1.
\end{aligned} \tag{5.14}$$

We now select a generic objective function as suggested by [29]:

$$\min \quad \sum_{j\in J}\sum_{k\in K} \beta_j^k + \lambda = -1 \tag{5.15}$$

[29] proposes to switch the roles of the objective function (5.15) and the normalization condition of polyhedron (5.14). See Appendix A.

This yields to the separation problem:

Separation problem

$$\begin{aligned}
\min_{\alpha,\beta,\gamma,\lambda} \quad & \sum_{j\in J}\sum_{k\in K} \beta_j^k q_j^{k^*} + \lambda\theta^* \\
\text{s.t} \quad & \sum_{k\in K} \alpha_{\mathrm{i}}^k = 0 \quad \forall \mathrm{i} \in I, \\
& \alpha_{\mathrm{i}}^k + \beta_j^k + \sum_{l\in J: l\neq j} (C_{\mathrm{i}j}^k - C_{\mathrm{i}l}^k)\gamma_{jl}^k + \\
& \qquad \lambda \pi^k R_{\mathrm{i}j}^k \geq 0 \quad \forall \mathrm{i} \in I, \forall j \in J, \forall k \in K, \\
& \gamma_{jl}^k \leq 0 \quad \forall \mathrm{i} \in I, \forall j \in J, \forall k \in K, \\
& \lambda \leq 0, \\
& \sum_{j\in J}\sum_{k\in K} \beta_j^k + \lambda = -1.
\end{aligned} \tag{5.16}$$

With this, the cuts to be added to master problem are:

$$0 \leq \sum_{j\in J}\sum_{k\in K} \beta_j^{kg} q_j^k + \lambda\theta \quad \forall g \in \text{extreme points}. \tag{5.17}$$

5.2.3 x remains in master problem, normalized

[19] applied standard Benders decomposition and got the following separation problem:

Standard k separation problem

$$\begin{aligned}
\min_{\alpha,\beta,\gamma,\delta} \quad & \alpha^k + \sum_{\mathrm{i}\in I} \gamma_{\mathrm{i}}^k x_{\mathrm{i}} \\
\text{s.t} \quad & \alpha^k - \beta_j^k \geq 0 \quad \forall j \in J, \\
& \beta_j^k + \gamma_{\mathrm{i}}^k + \sum_{l\in J: l\neq j} (C_{\mathrm{i}l}^k - C_{\mathrm{i}j}^k)\delta_{lj}^k \geq R_{\mathrm{i}j}^k \quad \forall \mathrm{i} \in I, \forall j \in J, \\
& \delta_{lj}^k \geq 0 \quad \forall l, j \in J : l \neq j.
\end{aligned} \tag{5.18}$$

We now apply the same as in §5.2.2 and obtain the following normalized separation problem:

Normalized k separation problem

$$\begin{aligned}
\min_{\alpha,\beta,\gamma,\delta,\lambda} \quad & \alpha^k + \sum_{\mathrm{i}\in I} \gamma_{\mathrm{i}}^k x_{\mathrm{i}} + \lambda^k \theta^k \\
\text{s.t} \quad & \alpha^k - \beta_j^k \geq 0 \quad \forall j \in J, \\
& \beta_j^k + \gamma_{\mathrm{i}}^k + \sum_{l\in J: l\neq j} (C_{\mathrm{i}l}^k - C_{\mathrm{i}j}^k)\delta_{lj}^k + \lambda^k R_{\mathrm{i}j}^k \geq 0 \quad \forall \mathrm{i} \in I, \forall j \in J, \\
& \sum_{\mathrm{i}\in I} \gamma_{\mathrm{i}}^k + \lambda^k = -1, \\
& \lambda^k \leq 0, \\
& \delta_{lj}^k \geq 0 \quad \forall l, j \in J : l \neq j.
\end{aligned} \tag{5.19}$$

The cuts to be added to the master problem are:

$$0 \leq \alpha^{kg^*} + \sum_{\mathrm{i}\in I} \gamma_{\mathrm{i}}^{kg^*} x_{\mathrm{i}} + \lambda^{kg^*} \theta^k \quad \forall g \in \text{extreme points}, \forall k \in K. \tag{5.20}$$

5.2.4 x and q remains in master problem, normalized

[19] applied standard Benders decomposition and got the following separation problem:

Standard k separation problem

$$\begin{aligned}
\min_{\alpha,\beta,\gamma} \quad & \sum_{j\in J} \alpha_j^k q_j^k + \sum_{\mathrm{i}\in I} \beta_{\mathrm{i}}^k x_{\mathrm{i}} \\
\text{s.t} \quad & \alpha_j^k + \beta_{\mathrm{i}}^k + \sum_{l\in J: l\neq j} (C_{\mathrm{i}l}^k - C_{\mathrm{i}j}^k)\gamma_{lj}^k \geq R_{\mathrm{i}j}^k \quad \forall \mathrm{i} \in I, \forall j \in J, \\
& \gamma_{lj}^k \geq 0 \quad \forall l, j \in J : l \neq j.
\end{aligned} \tag{5.21}$$

We now apply the same as in §5.2.2 and obtain the following normalized separation problem:

Normalized k separation problem

$$\begin{aligned}
\min_{\alpha,\beta,\gamma,\lambda} \quad & \sum_{j\in J} \alpha_j^k q_j^k + \sum_{\mathrm{i}\in I} \beta_{\mathrm{i}}^k x_{\mathrm{i}} + \lambda^k \theta^k \\
\text{s.t} \quad & \alpha_j^k + \beta_{\mathrm{i}}^k + \sum_{l\in J: l\neq j} (C_{\mathrm{il}}^k - C_{\mathrm{ij}}^k)\gamma_{lj}^k + \lambda^k R_{\mathrm{ij}}^k \geq 0 \quad \forall \mathrm{i}\in I, \forall j \in J, \\
& \sum_{j\in J} \alpha_j^k + \sum_{\mathrm{i}\in I} \beta_{\mathrm{i}}^k + \lambda^k = -1, \\
& \gamma_{lj}^k \geq 0 \quad \forall l, j \in J : l \neq j, \\
& \lambda^k \leq 0.
\end{aligned} \tag{5.22}$$

The cuts to be added to the master problem are:

$$0 \leq \sum_{j\in J} \alpha_j^{kg^*} q_j^{kg} + \sum_{\mathrm{i}\in I} \beta_{\mathrm{i}}^{kg^*} x_{\mathrm{i}} + \lambda^{kg^*} \theta^k \quad \forall g \in \text{extreme points}, \forall k \in K. \tag{5.23}$$

5.3 Security Stackelberg games

The approaches of this section are derived from formulation (MIP-p-S$_{c,q,y}$), which is the thightest SSG formulation found in the previous chapter:

$$\begin{aligned}
\max_{c,q,y} \quad & \sum_{j\in J}\sum_{k\in K} \left\{\pi^k (D^k(j|c) y_{jj}^k + D^k(j|u)(q_j^k - y_{jj}^k))\right\} \\
\text{s.t} \quad & \sum_{j\in J} q_j^k = 1 \qquad \forall k \in K, \\
& q_j^k \in \{0,1\} \qquad \forall j \in J, \forall k \in K, \\
& \sum_{j\in J} y_{lj}^k = c_l \qquad \forall l \in J, \forall k \in K, \\
& \sum_{l\in J} y_{lj}^k \leq m q_j^k \qquad \forall j \in J, \forall k \in K, \\
& 0 \leq y_{lj}^k \leq q_j^k \qquad \forall l, j \in J, \forall k \in K, \\
& A^k(j|c) y_{jj}^k + A^k(j|u)(q_j^k - y_{jj}^k) - \\
& A^k(l|c) y_{lj}^k - A^k(l|u)(q_j^k - y_{lj}^k) \geq 0 \qquad \forall j, l \in J, \forall k \in K.
\end{aligned} \tag{5.24}$$

5.3.1 q remains in the master problem

The first Benders decomposition approach derived is when q remains in master problem, and c and y are sent to subproblem.

Recall that subproblem contains the constraints where the variables c or y appear. Additionally, it contains the parts of the objective function where the same variables are, leaving in the master problem the parts containing exclusively x variables.

Subproblem

$$
\begin{aligned}
\max_{c,\,y} \quad & \sum_{j\in J}\sum_{k\in K}\pi^k(D^k(j|c)y^k_{jj} - D^k(j|u)y^k_{jj}) \\
\text{s.t} \quad & \sum_{j\in J} y^k_{lj} - c_l = 0, \quad \forall l \in J, \forall k \in K \quad (\alpha^k_l), \\
& \sum_{l\in J} y^k_{lj} \le m q^k_j, \quad \forall j \in j, \forall k \in K \quad (\beta^k_j), \\
& y^k_{lj} \le q^k_j, \quad \forall l, j \in J, \forall k \in K \quad (\gamma^k_{lj}), \\
& (A^k(j|c) - A^k(j|u))y^k_{jj} + (A^k(l|u) - A^k(l|c))y^k_{lj} \ge \\
& (A^k(l|u) - A^k(j|u))q^k_j \quad \forall l, j \in J, \forall k \in K \quad (\mu^k_{lj}), \\
& y^k_{lj} \ge 0 \quad \forall l, j \in J, \forall k \in K.
\end{aligned}
\tag{5.25}
$$

Separation problem

This consists of the dual of subproblem:

$$
\begin{aligned}
\min_{\alpha,\,\beta,\,\gamma,\,\mu} \quad & m\sum_{j\in J}\sum_{k\in K} q^k_j\beta^k_j + \sum_{j\in J}\sum_{l\in J}\sum_{k\in K}\{q^k_j\gamma^k_{lj} + (A^k(j|u) - A^k(l|u))q^k_j\mu^k_{lj}\} \\
\text{s.t} \quad & \sum_{k\in K}\alpha^k_l = 0 \quad \forall l \in J, \\
& \alpha^k_l + \beta^k_j + \gamma^k_{lj} + (A^k(l|c) - A^k(l|u))\mu^k_{lj} \ge 0 \quad \forall l, j \in J : l \ne j, \forall k \in K, \\
& \alpha^k_j + \beta^k_j + \gamma^k_{jj} + \sum_{l\in J: l\ne j}(A^k(j|u) - A^k(j|c))\mu^k_{lj} \ge \\
& \pi^k(D^k(j|c) - D^k(j|u)) \quad \forall j \in J, \forall k \in K, \\
& \beta^k_j \ge 0 \quad \forall j \in J, \forall k \in K, \\
& \gamma^k_{jl} \ge 0 \quad \forall l, j \in J, \forall k \in K, \\
& \mu^k_{jl} \ge 0 \quad \forall l, j \in J, \forall k \in K.
\end{aligned}
\tag{5.26}
$$

We obtain the following master problem which includes all cuts:

Reduced master problem

$$
\begin{aligned}
\max_{q,\theta} \quad & \theta + \sum_{j\in J}\sum_{k\in K} \pi^k D^k(j|u) q_j^k \\
\text{s.t} \quad & \sum_{j\in J} q_j^k = 1 \quad \forall k \in K, \\
& q_j^k \in \{0,1\} \quad \forall j \in J, \forall k \in K, \\
& \theta \le m \sum_{j\in J}\sum_{k\in K} q_j^k \beta_j^{kg} + \\
& \sum_{j\in J}\sum_{l\in J}\sum_{k\in K} \{q_j^k \gamma_{lj}^{kg} + (A^k(j|u) - A^k(l|u)) q_j^k \mu_{lj}^{kg}\} \quad \forall g \in \text{extreme points}, \\
& 0 \le m \sum_{j\in J}\sum_{k\in K} q_j^k \delta_j^{kh} + \\
& \sum_{j\in J}\sum_{l\in J}\sum_{k\in K} \{q_j^k \nu_{lj}^{kh} + (A^k(j|u) - A^k(l|u)) q_j^k \eta_{lj}^{kh}\} \quad \forall h \in \text{extreme rays}.
\end{aligned}
\tag{5.27}
$$

But, for our algorithms structure, we need to have a different objective function of master problem. So, we move the second term of the objective function to the optimality cuts. This yields the following cuts to be added to master problem:

$$
\begin{aligned}
\theta \le & \sum_{j\in J}\sum_{k\in K} \pi^k D^k(j|u) q_j^k + m \sum_{j\in J}\sum_{k\in K} q_j^k \beta_j^{kg^*} + \\
& \sum_{j\in J}\sum_{l\in J}\sum_{k\in K} \{q_j^k \gamma_{lj}^{kg^*} + (A^k(j|u) - A^k(l|u)) q_j^k \mu_{lj}^{kg^*}\} \quad \forall g \in \text{extreme points},
\end{aligned}
\tag{5.28}
$$

$$
\begin{aligned}
0 \le & \, m \sum_{j\in J}\sum_{k\in K} q_j^k \delta_j^{kh^*} + \\
& \sum_{j\in J}\sum_{l\in J}\sum_{k\in K} \{q_j^k \nu_{lj}^{kh^*} + (A^k(j|u) - A^k(l|u)) q_j^k \eta_{lj}^{kh^*}\} \quad \forall h \in \text{extreme rays}.
\end{aligned}
\tag{5.29}
$$

5.3.2 q remains in the master problem, normalized

The second Benders decomposition approach is an extension of the previous and analogous to the general setting, we apply the same normalization procedure described in [29] and [30].

From the previous approach, we know that the cut generation procedure is the following:

We know that for generating Benders cuts, we first solve the separation problem. If this problem is unbounded, we generate the following feasibility cut

$$
0 \le m \sum_{j\in J}\sum_{k\in K} q_j^k \delta_j^{k^*} + \sum_{j\in J}\sum_{l\in J}\sum_{k\in K} \{q_j^k \nu_{lj}^{k^*} + (A^k(j|u) - A^k(l|u)) q_j^k \eta_{lj}^{k^*}\} \tag{5.30}
$$

Otherwise, if we get an optimal solution such that

$$\sum_{j\in J}\sum_{k\in K}\pi^k D^k(j|u)q_j^{k^*} + m\sum_{j\in J}\sum_{k\in K} q_j^{k^*}\beta_j^{k^*} + \sum_{j\in J}\sum_{l\in J}\sum_{k\in K}\{q_j^{k^*}\gamma_{lj}^{k^*} + (A^k(j|u) - A^k(l|u))q_j^{k^*}\mu_{lj}^{k^*}\} < \theta^* \tag{5.31}$$

we generate the following optimality cut

$$\sum_{j\in J}\sum_{k\in K}\pi^k D^k(j|u)q_j^{k} + m\sum_{j\in J}\sum_{k\in K} q_j^{k}\beta_j^{k^*} + \sum_{j\in J}\sum_{l\in J}\sum_{k\in K}\{q_j^{k}\gamma_{lj}^{k^*} + (A^k(j|u) - A^k(l|u))q_j^{k}\mu_{lj}^{k^*}\} \geq \theta \tag{5.32}$$

So, we generate cuts if and only if the following problem is infeasible.

Cut generation problem

$$\begin{aligned}
&\sum_{j\in J}\sum_{k\in K}\pi^k(D^k(j|c)y_{jj}^k - D^k(j|u)y_{jj}^k) \geq \theta^* - \sum_{j\in J}\sum_{k\in K}\pi^k D^k(j|u)q_j^{k^*} \quad (\lambda),\\
&\sum_{j\in J} y_{lj}^k - c_l = 0, \quad \forall l \in J, \forall k \in K \quad (\alpha_l^k),\\
&\sum_{l\in J} y_{lj}^k \leq m q_j^{k^*}, \quad \forall j \in j, \forall k \in K \quad (\beta_j^k),\\
&y_{lj}^k \leq q_j^{k^*}, \quad \forall l, j \in J, \forall k \in K \quad (\gamma_{lj}^k),\\
&(A^k(j|c) - A^k(j|u))y_{jj}^k + (A^k(l|u) - A^k(l|c))y_{lj}^k \geq\\
&(A^k(l|u) - A^k(j|u))q_j^k \quad \forall l, j \in J, \forall k \in K \quad (\mu_{lj}^k),\\
&y_{lj}^k \geq 0 \quad \forall l, j \in J, \forall k \in K.
\end{aligned} \tag{5.33}$$

Which is the same that unbounded dual:

Dual cut generation problem

$$
\begin{aligned}
\min_{\alpha,\beta,\gamma,\mu,\lambda} \quad & m\sum_{j\in J}\sum_{k\in K} q_j^{k^*}\beta_j^k + \\
& \sum_{j\in J}\sum_{l\in J}\sum_{k\in K}\{q_j^{k^*}\gamma_{lj}^k + (A^k(l|u) - A^k(j|u))q_j^{k^*}\mu_{lj}^k\} + \lambda(\theta^* - \sum_{j\in J}\sum_{k\in K}\pi^k D^k(j|u)q_j^{k^*}) \\
\text{s.t} \quad & \sum_{k\in K}\alpha_l^k = 0 \quad \forall l \in J, \\
& \alpha_l^k + \beta_j^k + \gamma_{lj}^k + \\
& (A^k(l|u) - A^k(l|c))\mu_{lj}^k \geq 0 \quad \forall l, j \in J : l \neq j, \forall k \in K, \\
& \alpha_l^k + \beta_j^k + \gamma_{lj}^k + \sum_{l\in J: l\neq j}(A^k(j|c) - A^k(j|u))\mu_{lj}^k + \\
& \lambda\pi^k(D^k(j|c) - D^k(j|u)) \geq 0 \quad \forall j \in J, \forall k \in K, \\
& \beta_j^k \geq 0 \quad \forall j \in J, \forall k \in K, \\
& \gamma_{jl}^k \geq 0 \quad \forall l, j \in J, \forall k \in K, \\
& \mu_{jl}^k \leq 0 \quad \forall l, j \in J, \forall k \in K, \\
& \lambda \leq 0.
\end{aligned}
\tag{5.34}
$$

The cut will be:

$$
\begin{aligned}
0 \leq m\sum_{j\in J}\sum_{k\in K} q_j^k\beta_j^k + \sum_{j\in J}\sum_{l\in J}\sum_{k\in K}\{q_j^k\gamma_{lj}^k + (A^k(l|u) - A^k(j|u))q_j^k\mu_{lj}^k\} + \\
\lambda(\theta - \sum_{j\in J}\sum_{k\in K}\pi^k D^k(j|u)q_j^k)
\end{aligned}
\tag{5.35}
$$

We now apply the procedure described in [29] and [30].

Finding unboundedness in (5.34) consists of finding the set of extreme rays that satisfy:

$$
\begin{aligned}
m\sum_{j\in J}\sum_{k\in K} q_j^{k^*}\beta_j^k + \sum_{j\in J}\sum_{l\in J}\sum_{k\in K}\{q_j^{k^*}\gamma_{lj}^k + (A^k(l|u) - A^k(j|u))q_j^{k^*}\mu_{lj}^k\} + \\
\lambda(\theta^* - \sum_{j\in J}\sum_{k\in K}\pi^k D^k(j|u)q_j^{k^*}) < 0,
\end{aligned}
\tag{5.36}
$$

this because if the objective function of this problem is less than 0, this function goes to $-\infty$. But if we could fix the objective function to a negative value different to this, the problem would still be a unboundedness problem, but represented as an optimality problem. We now set the optimal value to:

$$m\sum_{j\in J}\sum_{k\in K} q_j^{k^*}\beta_j^k + \sum_{j\in J}\sum_{l\in J}\sum_{k\in K}\{q_j^{k^*}\gamma_{lj}^k + (A^k(l|u) - A^k(j|u))q_j^{k^*}\mu_{lj}^k\}+ \\ \lambda(\theta^* - \sum_{j\in J}\sum_{k\in K}\pi^k D^k(j|u)q_j^{k^*})^* = -1. \tag{5.37}$$

So now we can find extreme points that represent extreme rays.

With this, we get the alternative polyhedron:

Alternative polyhedron

$$\begin{aligned}
&\sum_{k\in K}\alpha_l^k = 0 \quad \forall l \in J,\\
&\alpha_l^k + \beta_j^k + \gamma_{lj}^k+\\
&(A^k(l|u) - A^k(l|c))\mu_{lj}^k \geq 0 \quad \forall l,j \in J : l \neq j, \forall k \in K,\\
&\alpha_l^k + \beta_j^k + \gamma_{lj}^k + \sum_{l\in J: l\neq j}(A^k(j|c) - A^k(j|u))\mu_{lj}^k+\\
&\lambda\pi^k(D^k(j|c) - D^k(j|u)) \geq 0 \quad \forall j \in J, \forall k \in K,\\
&m\sum_{j\in J}\sum_{k\in K} q_j^{k^*}\beta_j^k + \sum_{j\in J}\sum_{l\in J}\sum_{k\in K}\{q_j^{k^*}\gamma_{lj}^k + (A^k(l|u) - A^k(j|u))q_j^{k^*}\mu_{lj}^k\}+\\
&\lambda(\theta^* - \sum_{j\in J}\sum_{k\in K}\pi^k D^k(j|u)q_j^{k^*}) = -1,\\
&\beta_j^k \geq 0 \quad \forall j \in J, \forall k \in K,\\
&\gamma_{jl}^k \geq 0 \quad \forall l,j \in J, \forall k \in K,\\
&\mu_{jl}^k \leq 0 \quad \forall l,j \in J, \forall k \in K\\
&\lambda \leq 0.
\end{aligned} \tag{5.38}$$

We now apply the same procedure than §5.2.2, and obtain the separation problem by switching roles between objective function and normalization condition:

Separation problem

$$
\begin{aligned}
\min_{\alpha,\beta,\gamma,\mu,\lambda} \quad & m\sum_{j\in J}\sum_{k\in K} q_j^{k^*}\beta_j^k + \\
& \sum_{j\in J}\sum_{l\in J}\sum_{k\in K}\{q_j^{k^*}\gamma_{lj}^k + (A^k(j|u) - A^k(l|u))q_j^{k^*}\mu_{lj}^k\} + \\
& \lambda(\sum_{j\in J}\sum_{k\in K}\pi^k D^k(j|u)q_j^{k^*} - \theta^*) \\
\text{s.t} \quad & \sum_{k\in K}\alpha_l^k = 0 \quad \forall l \in J, \\
& \alpha_l^k + \beta_j^k + \gamma_{lj}^k + \\
& (A^k(l|c) - A^k(l|u))\mu_{lj}^k \geq 0 \quad \forall l,j \in J : l \neq j, \forall k \in K, \\
& \alpha_l^k + \beta_j^k + \gamma_{lj}^k + \sum_{l\in J: l\neq j}(A^k(j|u) - A^k(j|c))\mu_{lj}^k + \\
& \lambda\pi^k(D^k(j|u) - D^k(j|c)) \geq 0 \quad \forall j \in J, \forall k \in K, \\
& \beta_j^k \geq 0 \quad \forall j \in J, \forall k \in K, \\
& \gamma_{jl}^k \geq 0 \quad \forall l,j \in J, \forall k \in K, \\
& \mu_{jl}^k \geq 0 \quad \forall l,j \in J, \forall k \in K, \\
& \lambda \geq 0, \\
& \sum_{j\in J}\sum_{k\in K}\beta_j^k + \sum_{j\in J}\sum_{l\in J}\sum_{k\in K}(\gamma_{lj}^k - \mu_{lj}^k) - \lambda = -1.
\end{aligned}
\tag{5.39}
$$

The cuts to be added to master problem are:

$$
\begin{aligned}
0 \leq m\sum_{j\in J}\sum_{k\in K} q_j^k\beta_j^{kg^*} + & \sum_{j\in J}\sum_{l\in J}\sum_{k\in K}\{q_j^k\gamma_{lj}^{kg^*} + (A^k(j|u) - A^k(l|u))q_j^k\mu_{lj}^{kg^*}\} - \\
& \lambda^*(\theta - \sum_{j\in J}\sum_{k\in K}\pi^k D^k(j|u)q_j^k) \quad \forall g \in \text{extreme points.}
\end{aligned}
\tag{5.40}
$$

5.3.3 c remains in master problem, normalized

[19] applied standard Benders decomposition and got the following separation problem:

Standard k separation problem

$$
\begin{aligned}
\min_{\alpha,\beta,\gamma,\delta,\varepsilon} \quad & \sum_{l\in J}\beta_l^k c_l + \alpha^k \\
\text{s.t} \quad & -m\gamma_j^k - \sum_{l\in J}\delta_{lj}^k + \alpha^k + \\
& \sum_{l\in J}(A^k(l|u) - A^k(j|u))\varepsilon_{lj}^k \geq D^k(j|u) \quad \forall j \in J, \\
& \beta_j^k + \gamma_j^k + \delta_{jj}^k + \\
& \sum_{l\in J: l\neq j}(A^k(j|u) - A^k(j|c))\varepsilon_{lj}^k \geq D^k(j|c) - D^k(j|u) \quad \forall j \in J, \\
& \beta_l^k + \gamma_j^k + \delta_{lj}^k + (A^k(l|c) - A^k(l|u))\varepsilon_{lj}^k \geq 0 \quad \forall l, j \in J : l \neq j, \\
& \gamma_j^k \geq 0 \quad \forall j \in J, \\
& \delta_{lj}^k \geq 0 \quad \forall l, j \in J, \\
& \varepsilon_{lj}^k \geq 0 \quad \forall l, j \in J.
\end{aligned}
\tag{5.41}
$$

We now apply the same as in §5.3.2 and obtain the following normalized separation problem:

Normalized k separation problem

$$
\begin{aligned}
\min_{\alpha,\beta,\gamma,\delta,\varepsilon,\lambda} \quad & \sum_{l\in J}\beta_l^k c_l + \alpha^k + \lambda^k\theta^k \\
\text{s.t} \quad & -m\gamma_j^k - \sum_{l\in J}\delta_{lj}^k + \alpha^k + \\
& \sum_{l\in J}(A^k(l|u) - A^k(j|u))\varepsilon_{lj}^k + \lambda^k D^k(j|u) \geq 0 \quad \forall j \in J, \\
& \beta_j^k + \gamma_j^k + \delta_{jj}^k + \sum_{l\in J: l\neq j}(A^k(j|u) - A^k(j|c))\varepsilon_{lj}^k + \\
& \lambda^k(D^k(j|c) - D^k(j|u)) \geq 0 \quad \forall j \in J, \\
& \beta_l^k + \gamma_j^k + \delta_{lj}^k + (A^k(l|c) - A^k(l|u))\varepsilon_{lj}^k \geq 0 \quad \forall l, j \in J : l \neq j, \\
& \sum_{l\in J}\beta_l^k + \lambda^k = -1, \\
& \gamma_j^k \geq 0 \quad \forall j \in J, \\
& \delta_{lj}^k \geq 0 \quad \forall l, j \in J, \\
& \varepsilon_{lj}^k \geq 0 \quad \forall l, j \in J, \\
& \lambda^k \leq 0.
\end{aligned}
\tag{5.42}
$$

The cuts to be added to master problem are:

$$
0 \leq \sum_{l\in J}\beta_l^{kg^*} c_l + \alpha^{kg^*} + \lambda^{kg^*}\theta^k \quad \forall g \in \text{extreme points}, \forall k \in K. \tag{5.43}
$$

5.3.4 c and q remains in master problem, normalized

[19] applied standard Benders decomposition and got the following separation problem:

Standard k separation problem

$$\begin{aligned}
\min_{\alpha,\beta,\gamma,\delta} \quad & \sum_{l\in J}\delta_l^k c_l + m\sum_{j\in J}\alpha_j^k q_j^k + \sum_{l\in J}\sum_{j\in J}\{\beta_{lj}^k q_j^k + (A^k(j|u) - A^k(l|u))\gamma_{lj}^k q_j^k\} \\
\text{s.t} \quad & \delta_l^k + \alpha_j^k + \beta_{lj}^k + (A^k(l|c) - A^k(l|u))\gamma_{lj}^k \geq 0 \quad \forall l,j \in J : l \neq j, \\
& \delta_j^k + \alpha_j^k + \beta_{jj}^k + (A^k(j|u) - A^k(j|c))\sum_{l\in J:l\neq j}\gamma_{lj}^k \geq \\
& (D^k(j|c) - D^k(j|u)) \quad \forall j \in J, \\
& \alpha_j^k \geq 0 \quad \forall j \in J \\
& \beta_{lj}^k \geq 0 \quad \forall l,j \in J, \\
& \gamma_{lj}^k \geq 0 \quad \forall l,j \in J.
\end{aligned} \tag{5.44}$$

We now apply the same as in §5.3.2 and obtain the following normalized separation problem:

Normalized k separation problem

$$\begin{aligned}
\min_{\alpha,\beta,\gamma,\delta,\lambda} \quad & \sum_{l\in J}\delta_l^k c_l + m\sum_{j\in J}\alpha_j^k q_j^k + \sum_{l\in J}\sum_{j\in J}\{\beta_{lj}^k q_j^k + (A^k(j|u) - A^k(l|u))\gamma_{lj}^k q_j^k\} \\
& + \lambda^k(\theta^k - \sum_{j\in J} D^k(j|u)q_j^k) \\
\text{s.t} \quad & \delta_l^k + \alpha_j^k + \beta_{lj}^k + (A^k(l|c) - A^k(l|u))\gamma_{lj}^k \geq 0 \quad \forall l,j \in J : l \neq j, \\
& \delta_j^k + \alpha_j^k + \beta_{jj}^k + (A^k(j|u) - A^k(j|c))\sum_{l\in J:l\neq j}\gamma_{lj}^k + \\
& \lambda^k(D^k(j|c) - D^k(j|u)) \geq 0 \quad \forall j \in J, \\
& \sum_{l\in J}\delta_l^k + \sum_{j\in J}\alpha_j^k + \sum_{j\in J}\sum_{l\in J}(\beta_{lj}^k + \gamma_{lj}^k) + \lambda^k = -1, \\
& \alpha_j^k \geq 0 \quad \forall j \in J, \\
& \beta_{lj}^k \geq 0 \quad \forall l,j \in J, \\
& \gamma_{lj}^k \geq 0 \quad \forall l,j \in J, \\
& \lambda^k \leq 0.
\end{aligned} \tag{5.45}$$

The cuts to be added to master problem are:

$$\begin{aligned}
0 \leq \sum_{l\in J}\delta_l^{kg^*} c_l + m\sum_{j\in J}\alpha_j^{kg^*} q_j^k + \sum_{l\in J}\sum_{j\in J}\{\beta_{lj}^{kg^*} q_j^k + (A^k(j|u) - A^k(l|u))\gamma_{lj}^{kg^*} q_j^k\} + \\
\lambda^{kg^*}(\theta^k - \sum_{j\in J} D^k(j|u)q_j^k) \quad \forall g \in \text{extreme points}, \forall k \in K.
\end{aligned} \tag{5.46}$$

Chapter 6

Branching algorithms

6.1 Introduction

In this chapter, we propose a few cutting plane schemes for the general and security Stackelberg games, due to [19].

The cut generation algorithms consist in strengthening the linear relaxation of fast and weak formulations, (D2$_{x,q,s,f}$) and (ERASER$_{c,q,s,f}$) derived in chapter 4, by using the cuts obtained from the Benders decomposition approaches of (MIP-p-G$_{x,q,z}$) and (MIP-p-S$_{c,q,z}$)-tight formulations- derived in chapter 5. Given that both formulations are equivalent, the respective cuts are valid for (D2$_{x,q,s,f}$) and (ERASER$_{c,q,s,f}$), and cut the polyhedron of its linear relaxation.

According to [19], the resulting formulations (D2$_{x,q,s,f}$)+{valid cuts} or (ERASER$_{c,q,s,f}$)+ {valid cuts} will have the same tight bound as the linear relaxations of (MIP-p-G$_{x,q,z}$) and (MIP-p-S$_{c,q,z}$), but will be significantly sparser in terms of variables and constraints, leading to an improved performance of the approach over the tightest formulation.

This cut generation is applied only in the root node or in the whole branching tree, depending on the type of branching algorithm. Recall that branching algorithms consist of explore the nodes of the branching tree with a branch & bound scheme described in §2.5. If the cut generation is applied in the root node and across the whole tree, in addition with the integer bounds, the algorithm is called branch & cut. If the cut generation is applied only in the rode node, and integer bounds are added across the tree, the algorithm is called cut & branch.

6.2 Branch & cut algorithm

In this thesis, we perform two branch & cut algorithms for general and security Stackelberg games respectively. The first performed algorithm consists of using the standard Benders decomposition approach with q in master problem for generating cuts, and the second one uses the normalized Benders decomposition approach keeping the same variable in master

problem.

We now draw the cutting planes algorithms that will be used in a

Before drawing the algorithms, we define some concepts:

MASTER: linear relaxation of (D2$_{x,q,s,f}$) or (ERASER$_{c,q,s,f}$), for general or security Stackelberg games, respectively.

v(MASTER): optimal value of *MASTER*.

SEPARATION PROBLEM: separation problems for different decomposition approaches in chapter 5.

Standard Benders decomposition approach

In this section, we outline the cut generation algorithm that uses as separation problem the standard Benders decomposition approach with q in master problem. Algorithm 2 can be used in general or security setting.

Algorithm 2: Cut generation algorithm for standard Benders decomposition approach.

```
Data: Game data
Result: Integer solution
initialization;
UB := +∞;
DualLB := −∞;
PrimalLB := CPLEX.Incumbent;
while UB − PrimalLB > εUB do
    Solve MASTER;
    Identify optimal MASTER fractional solution;
    Update UB=v(MASTER);
    while UB − DualLB > εUB do
        Feed solution into SEPARATION PROBLEM;
        Solve SEPARATION PROBLEM;
        if SEPARATION PROBLEM is unbounded then
            Generate feasibility cut (5.6) or (5.30);
        else
            Generate optimality cut (5.8) or (5.32);
        end
        if ∃ generated cuts which are violated then
            Add cuts to MASTER;
        end
        Update DualLB to objective value of SEPARATION PROBLEM;
    end
end
Finish CPLEX solve;
```

From line 5, the algorithm 2 shows the cut generation procedure that is applied across the whole branch & bound search tree. This algorithm specifies the conditions for generating feasibility and optimality cuts and the stopping criteria. Algorithm 2 is used inside the algorithm described in [23], in the cut generation step.

Normalized Benders decomposition approach

In this section, we outline the cut generation algorithm that uses as separation problem the normalized Benders decomposition approach with q in master problem. Algorithm 3 can be used in general or security setting.

Algorithm 3: Cut generation algorithm for normalized Benders decomposition approach.

```
   Data: Game data
   Result: Integer solution
 1 initialization;
 2 UB := +∞;
 3 DualLB := −∞;
 4 PrimalLB := CPLEX.Incumbent;
 5 while UB − PrimalLB > εUB do
 6 |   Solve MASTER;
 7 |   Identify optimal MASTER fractional solution;
 8 |   Update UB=v(MASTER);
 9 |   while UB − DualLB > εUB do
10 |   |   Feed solution into SEPARATION PROBLEM;
11 |   |   Solve SEPARATION PROBLEM;
12 |   |   if SEPARATION PROBLEM is optimal then
13 |   |   |   Generate normalized cut (5.17) or (5.40);
14 |   |   end
15 |   |   Add cuts to MASTER;
16 |   |   Update DualLB to objective value of SEPARATION PROBLEM;
17 |   end
18 end
19 Finish CPLEX solve;
```

Algorithm 3 also generates cuts in the whole search tree, but only one type of cut, the normalized cut. This algorithm specifies that, if exists, the normalized cut is generated, not having to be violated for doing it. Algorithm 3 is used inside the algorithm described in [23], in the cut generation step.

6.3 Cut & branch algorithm

In this thesis, we perform two cut & branch algorithms for general and security Stackelberg games respectively. The first algorithm consists of using normalized Benders decomposition approaches with x or c in master problem, and the second uses normalized Benders decomposition approaches with x and q or c and q, respectively. It is important to mention that

cut & branch algorithms for the same approaches but in standard Benders decomposition were already developed in [19].

We now outline the cut generation algorithm for the four situations explained in the previous paragraph. Algorithm 4 can be used in general and security setting.

Algorithm 4: Cut generation algorithm for normalized Benders decomposition approach.

```
   Data: Game data
   Result: Integer solution
 1 initialization;
 2 UB := +∞;
 3 DualLB := −∞;
 4 PrimalLB := CPLEX.Incumbent;
 5 while UB − PrimalLB > εUB do
 6 |   Solve MASTER;
 7 |   Identify optimal MASTER fractional solution;
 8 |   Update UB=v(MASTER);
 9 |   if Node=root then
10 |   |   while UB − DualLB > εUB do
11 |   |   |   Feed solution into SEPARATION PROBLEM;
12 |   |   |   Solve SEPARATION PROBLEM;
13 |   |   |   if SEPARATION PROBLEM is optimal then
14 |   |   |   |   Generate normalized cut of (5.20), (5.23), (5.43) or (5.46);
15 |   |   |   end
16 |   |   |   Add cuts to MASTER;
17 |   |   |   Update DualLB to objective value of SEPARATION PROBLEM;
18 |   |   end
19 |   end
20 end
21 Finish CPLEX solve;
```

Algorithm 4 shows that the cut generation procedure is applied only in root node, that's specified in line 9. Algorithm 3 is used inside the algorithm described in [23], in the cut generation step.

6.4 Implementation

When implementing branching algorithms for general and security Stackelberg games, [19] found some useful heuristics that improved performance in some of them. For completeness, we briefly describe each of them. For detailed discussion see [19].

6.4.1 Cut loop stabilization

The purpose of a cut loop stabilization procedure is reducing the number of times we have to solve the separation problems before to conclude solving the problem. The classical cut

generation scheme is known to have a very slow convergence. [28] point out that the convergence performance of the cut strategy depends on the point chosen to be separated in each iteration, therefore the performance of the cut loop can be improved using a simple in-out stabilization procedure like the one shown in [9]. The heuristic procedure is simple: we first get a fractional solution from solving the master problem (a_{out}), and a feasible solution for the original MILP (a_{in}). In the standard cut generation scheme, we feed the a_{out} to the separation problems, but with this cut loop stabilization heuristics, we feed instead and intermediate point:

$$a_{sep} = \lambda a_{out} + (1 - \lambda) a_{in} \tag{6.1}$$

where $0 < \lambda \leq 1$.

It's easy to see that $\lambda = 1$ implies no stabilization.

This heuristic will tried in most of instances.

6.4.2 Initial feasible solution

This heuristic is based in [47], where they limit the possible leader mixed strategies to choose, by using s-uniform strategies. s-uniform strategies are those where the probability of playing a given pure strategy is a multiple of $1/s$, for some integer s. They solve for this strategy a formulation similar to (DOBSS) or (SDOBSS) respectively, where the x_{i} or c_j variables are forced to take values in the set $\{0, 1/s, 2/s, \ldots, 1\}$, through the use of auxiliary integer variables $r_{\mathrm{i}} \in \{0, 1, 2, \ldots, s\}$ such that $s x_{\mathrm{i}} = r_{\mathrm{i}}$.

[19] applies the same strategy, but using (MIP-p-G$_{x,q,z}$) and (MIP-p-S$_{c,q,y}$) instead of (DOBSS) and (SDOBSS), due to its enhanced computational results. This solution returns an s-uniform leader mixed strategy which is feasible in GSG or SSG and gives a lower bound on the optimum value.

This heuristic will be applied in every single instance and algorithm, and is the only method we use to get the initial feasible solution.

6.4.3 Lower bound enhancing heuristics

[19] proposes two lower bound enhancing heuristics, basic and advanced. These are briefly described next.

Basic heuristic

Given a leader mixed strategy $\bar{x}$, from optimal solution of the master problem, the best response to $\bar{x}$ of each follower have to be computed. For doing this, we first have to compute expected utility for selecting each pure strategy $j \in J$:

$$FEU_j^k = \sum_{\mathrm{i} \in I} C_{\mathrm{ij}}^k \bar{x}_{\mathrm{i}} \qquad \text{for general setting, or} \tag{6.2}$$

$$AEU_j^k = A^k(j|c)\bar{c}_j + A^k(j|u)(1-\bar{c}_j) \qquad \text{for security setting,} \tag{6.3}$$

and select the strategy $j(k)$ that maximizes each follower's expected utility over all the pure strategies $j \in J$:

$$j(k) = arg \max_{j \in J} FEU_j^k \qquad \text{for general setting, or} \tag{6.4}$$

$$j(k) = arg \max_{j \in J} AEU_j^k \qquad \text{for security setting.} \tag{6.5}$$

Then, construct the follower's strategy q^* such that $q_j^{k^*} = 1$ for $j \in J = j(k)$ and $q_j^{k^*} = 0$ for $j \in J : j \neq j(k)$. Then $(\bar{x}, q^*)$ (or $(\bar{c}, q^*)$ for security setting) is a feasible solution to the integer problem with objective value:

$$\sum_{\mathrm{i} \in I} \sum_{j \in J} \sum_{k \in K} \pi^k R_{\mathrm{i}j}^k \bar{x}_{\mathrm{i}} q_j^{k^*} \qquad \text{for general setting, or} \tag{6.6}$$

$$\sum_{j \in J} \sum_{k \in K} q_j^{k^*} \pi^k (D^k(j|c)\bar{c}_j + D^k(j|u)(1-\bar{c}_j)) \qquad \text{for security setting.} \tag{6.7}$$

Advanced heuristic

The advanced heuristic optimizes the leader strategy over all mixed strategies for which q^* from the basic heuristic is a best response, by solving the following problem for general Stackelberg games:

$$\begin{aligned}
\max \quad & \sum_{\mathrm{i} \in I} \sum_{k \in K} \pi^k R_{\mathrm{i}j(k)}^k x_{\mathrm{i}} \\
\text{s.t} \quad & \sum_{\mathrm{i} \in I} x_{\mathrm{i}} = 1, \\
& x_{\mathrm{i}} \geq 0 \qquad \forall \mathrm{i} \in I, \\
& \sum_{\mathrm{i} \in I} C_{\mathrm{i}j(k)} x_{\mathrm{i}} \geq \sum_{\mathrm{i} \in I} C_{\mathrm{i}j} x_{\mathrm{i}} \qquad \forall k \in K, \forall j \in J : j \neq j(k),
\end{aligned} \tag{6.8}$$

or the following for security Stackelberg games:

$$
\begin{aligned}
\max \quad & \sum_{k \in K} \pi^k (D^k(j(k)|c) c_{j(k)} + D^k(j(k)|u)(1 - c_{j(k)}) \\
\text{s.t} \quad & \sum_{j \in J} c_j \leq m, \\
& c_j \in [0,1] \qquad \forall j \in J, \\
& A^k(j(k)|c) c_{j(k)} + A^k(j(k)|u)(1 - c_{j(k)}) \geq \\
& A^k(j|c) c_j + A^k(j|u)(1 - c_j) \qquad \forall k \in K, \forall j \in J : j \neq j(k).
\end{aligned}
\tag{6.9}
$$

If this optimal solution (basic or advanced) is better than incumbent solution, then we have to update the incumbent solution and incumbent value.

6.4.4 Upper bound enhancing heuristics

This heuristic consists of selecting a subset of q variables and setting it to binary in the tight formulation (variables that are not binary in the current solution).

This heuristic can be used to enhance the quality of upper bound after solve the root node with any of the cutting plane approaches.

We next outline the algorithm, but we first need some definitions:

(rMIP-p-G$_{x,q,z}$): relaxation of (MIP-p-G$_{x,q,z}$), by setting a subset of the continuous q variables to binary.

(rMIP-p-S$_{c,q,y}$): relaxation of (MIP-p-S$_{c,q,y}$), by setting a subset of the continuous q variables to binary.

Algorithm 5: Upper bound heuristic.

Data: Fraccional solution from MASTER, current upper bound
Result: Upper bound cut or nothing

```
initialization;
if First time in node 1 then
    Construct (rMIP-p-G_{x,q,z}) or (rMIP-p-S_{c,q,y});
    Solve (rMIP-p-G_{x,q,z}) or (rMIP-p-S_{c,q,y});
    if ((rMIP-p-G_{x,q,z}) or (rMIP-p-S_{c,q,y})) < UB then
        Add cut to MASTER: sum_{k in K} pi^k theta^k <= ((rMIP-p-G_{x,q,z}) or (rMIP-p-S_{c,q,y})) ;
    end
end
```

Chapter 7

Computational experiments

7.1 Introduction

In this chapter, we compare the performance of the different solution methods across different sets of instances.

We first select a subset of our most competitive algorithms and their best heuristic configurations. After doing this, we compare them with the most competitive algorithms in [19].

7.2 Instances

Instances used to compare the algorithms performance are constructed according [19]. Each configuration is detailed in its respective section.

7.2.1 General Stackelberg games

Table (7.1) shows the sizes of the two sets of general instances to be evaluated in our experiments.

Set	|Leader pure strategies (I)|	|Follower pure strategies (J)|	|Followers (K)|
A	{5}	{5}	{25,30,...,95}
B	{5,6,7,8,9,10}	{5,6,7,8,9,10}	{23}

Table 7.1: Sizes of GSG instances to compare.

Set A fixes the number of pure strategies for leader and follower, and scales up the number of followers. Set B fixes the number of followers, and scales up the number of pure strategies for leader and follower.

Additionally, leader and follower payoff values are uniform randomly generated between 0 and 10. The probability distribution over follower types is generated randomly in $[0, 1]^{|K|}$.

The number of instances generated for each size is 5, which sums 75 instances in set A and 30 instances in set B.

7.2.2 Security Stackelberg games

Table (7.2) shows the main parameters of the two sets of security instances that will be evaluated.

Set	|Targets (J)|	|Sec. resources (m)|	|Attackers (K)|
A	{5}	50% $\|J\|$	{45,50,...,95}
B	{8,10,12,14}	50% $\|J\|$	{25}

Table 7.2: Sizes of SSG instances to compare.

Set A fixes the number of targets, and scales up the number of attackers. Set B fixes the number of attackers, and scales up the number of targets. Security resources are 50% of number of targets in both sets of instances. When the number of targets J is odd, the number of resources is the ceiling of its half.

Additionally, penalties for defender and attacker are randomly generated between 0 and 5, and rewards for both of them between 5 and 10. The probability distribution over attacker types is generated randomly in $[0, 1]^{|K|}$. The number of instances generated for each size is 5, which sums 55 instances in set A and 20 instances in set B.

7.3 Computational resources

The computational infrastructure used to run the experiments consists of 2×Intel® Xeon® E5-2660 V2, 2.20 GHz. This is equipped with 10 cores, 48 GB RAM and operating system CentOS Linux release 7.5.1804.

The programming language used for coding is Python 2.7 and the solver is CPLEX® Optimization Studio V12.8.

7.4 Best configuration

We compare four different algorithms for general and security setting. We evaluate the use of different heuristic configurations for the promising variants of the algorithms, according to §6.4: cut loop stabilization, lower bound enhancing heuristics and upper bound enhancing heuristics, in order to decide the best candidates with best heuristic configurations.

7.4.1 General Stackelberg games

In this section, we compare four general algorithms: branch & cut with q in master problem, normalized branch & cut with q in master problem, normalized cut & branch with x in master problem and normalized cut & branch with x and q in master problem.

We separate experiments by set of instances, in order to identify the best algorithm for each type of them separately.

Instances A

Table (7.3) shows the performance of four algorithms in the set of general instances A with no heuristic configurations. The whole set of 75 instances is considered for this.

Algorithm	Av. time	% solve
(B&C$_q$)	8061	40%
(B&C$_q$norm)	6496	50%
(C&B$_x$norm)	**4881**	**68%**
(C&B$_{x,q}$norm)	**4856**	**68%**

Table 7.3: Algorithms performance for set of general instances A without heuristic configurations.

We can see that the best performance is produced by (C&B$_x$norm) and (C&B$_{x,q}$norm), both normalized cut & branch algorithms. Next, we try some heuristic configurations to decide the best configuration for these algorithms.

For trying the different heuristic configurations, we select a subset of instances: $I = J = \{5\}$, $K = \{25, 30, \ldots, 80\}$.

Impact of cut loop stabilization Table (7.4) shows average results for set of general instances A using different stabilization parameters. $\lambda = 0.2$ means aggressive stabilization and $\lambda = 1$ means no stabilization. We can see that the best average time performance, in both algorithms, is produced without stabilization. The table even shows an inverse relation between stabilization and average solution time. For the first algorithm, $\lambda = 1$ solves the relaxation faster than others, but with a very little difference. The second algorithm performs much better with $\lambda = 1$, solving the relaxation in 65% of time of its nearest competitor. This could have a big influence in the better performance of this parameter in the average time. For the first algorithm, $\lambda = 1$ explores fewer number of nodes, but only a difference of 0.5% with its nearest competitor. This little difference, plus the little difference in the relaxation time, explain the little average time difference. For the second algorithm, $\lambda = 0.7$ explores the less number of nodes, but with a difference 0.5% against $\lambda = 1$ (which got the best relaxation time). This difference is not as important as the relaxation time difference, because it wasn't enough to get a better average time.

Figure (7.1) shows the percentage of problems solved as time goes by. Time limit is 10.800 seconds (3 hours). Grey line represents no stabilization. The graph shows the same as table (7.4): no stabilization outperforms other stabilization parameters for both algorithms, although the difference is very low, almost unreadable.

λ	(C&B$_x$norm)	(C&B$_{x,q}$norm)
Average time		
0.2	3506	3455
0.5	3455	3448
0.7	3440	3375
1	**3401**	**3370**
Average LP time		
0.2	29.54	20.83
0.5	29.35	18.75
0.7	28.33	16.8
1	**28.21**	**11.03**
Average nodes		
0.2	4854361	5379266
0.5	4854361	5235678
0.7	4854664	**4852219**
1	**4828434**	4870836

Table 7.4: Algorithms average time, average LP time and average nodes for set of general instances A comparing different stabilization parameters.

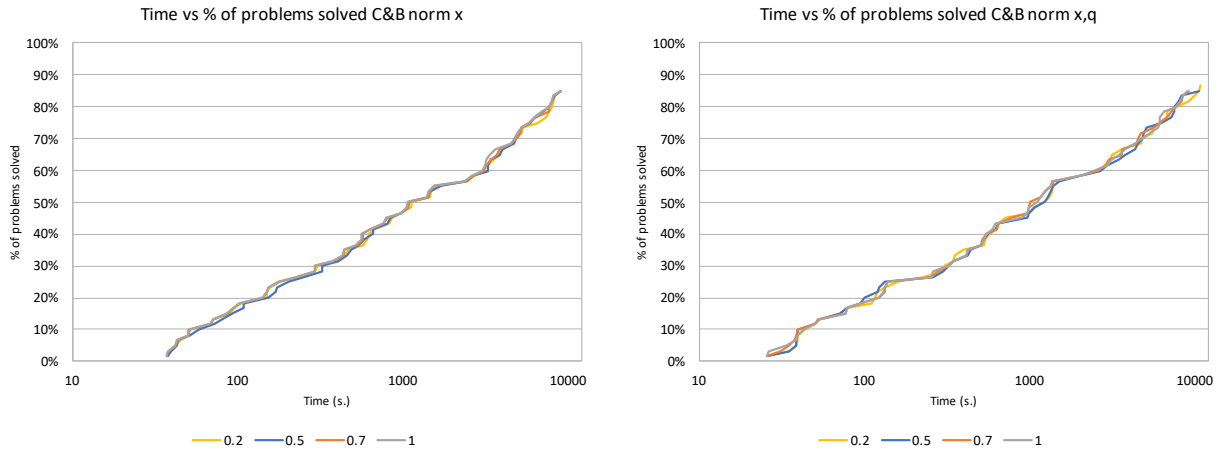

Figure 7.1: Effect of stabilization on solution time for set of general instances A.

With this analysis, we can conclude that the best stabilization parameter for (C&B$_x$norm) and (C&B$_{x,q}$norm) is $\lambda = 1$.

Impact of lower bound enhancing heuristics Table (7.5) shows the average results with different LB heuristic configurations for set of general instances A. We can see clearly that the best average time performance in both algorithms is produced with no LB heuristic configuration. We can see that LP time is better for basic LB heuristic, for both algorithms, but the difference is minimal between all configurations. We can conclude that the LB heuristic configurations doesn't have an impact in the root node. The advanced LB heuristic explores less nodes for both algorithms and, for the left algorithm, it does together with

basic and advanced heuristic. No LB heuristic explores the highest number of nodes for both algorithms.

Configuration	(C&B$_x$norm)	(C&B$_{x,q}$norm)
Average time		
No LB heuristic	**3401**	**3370**
Basic LB heuristic	3971	4022
Basic and advanced LB heuristic	4015	3915
Advanced LB heuristic	4048	3903
Average LP time		
No LB heuristic	28.2	11.03
Basic LB heuristic	**28.18**	**10.75**
Basic and advanced LB heuristic	28.21	10.77
Advanced LB heuristic	28.49	10.76
Average nodes		
No LB heuristic	4828434	4870836
Basic LB heuristic	4458280	4533744
Basic and advanced LB heuristic	**4287074**	4345168
Advanced LB heuristic	**4287074**	**4336773**

Table 7.5: Algorithms average time, average LP time and average nodes for set of general instances A comparing different parameters of LB heuristics.

Figure (7.2) shows the effect of lower bound heuristics on average solution time for each number of follower types. In this graph, we present average value of the five instances in each size. Again, we can see clearly that the best performance is produced for both algorithms with no LB heuristics, for every instance size.

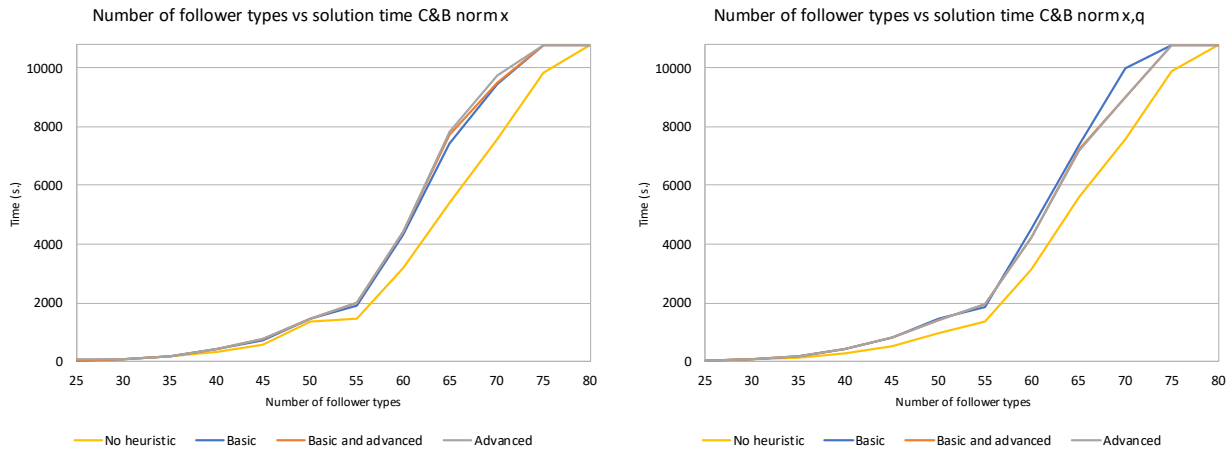

Figure 7.2: Effect of LB heuristics on average solution time for general instances A.

Figure (7.3) shows the effect of lower bound heuristics on average relaxation time for each number of follower types. In this graph, we present average value of the five instances in each

size. We can see that the lines are very close to each other across all the instances (we can see that scale of the graphs is 50 seconds). Is interesting to see that, for the second algorithm, the relaxation time is not necessarily increasing with the number of follower types.

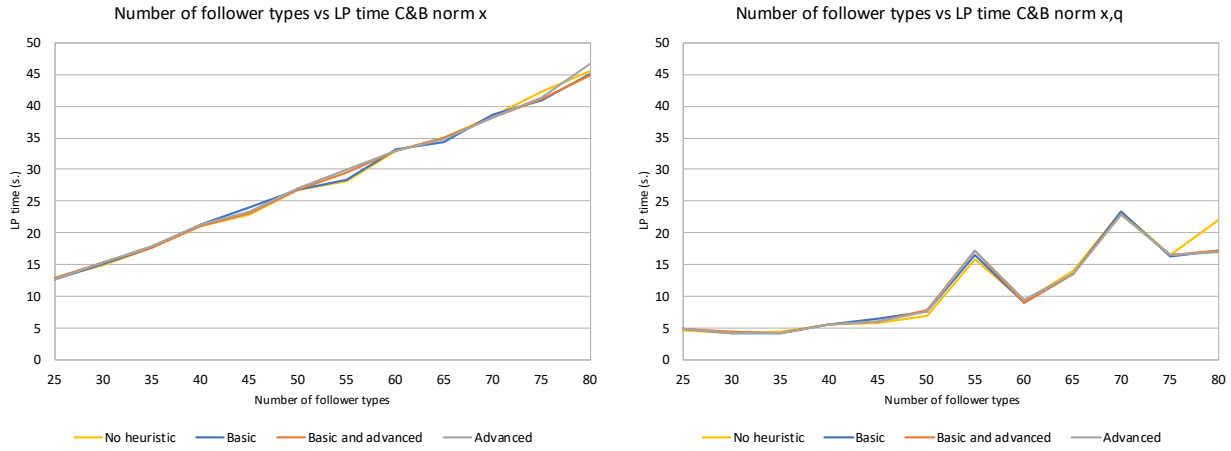

Figure 7.3: Effect of LB heuristics on average relaxation time for general instances A.

Figure (7.4) shows the effect of lower bound heuristics on average nodes explored for each number of follower types. In this graph, we present average value of the five instances in each size. We can see clearly a difference for no LB heuristic, which explores the highest number of nodes as the number of followers increases, but explores similar numbers than others in the bottom instances.

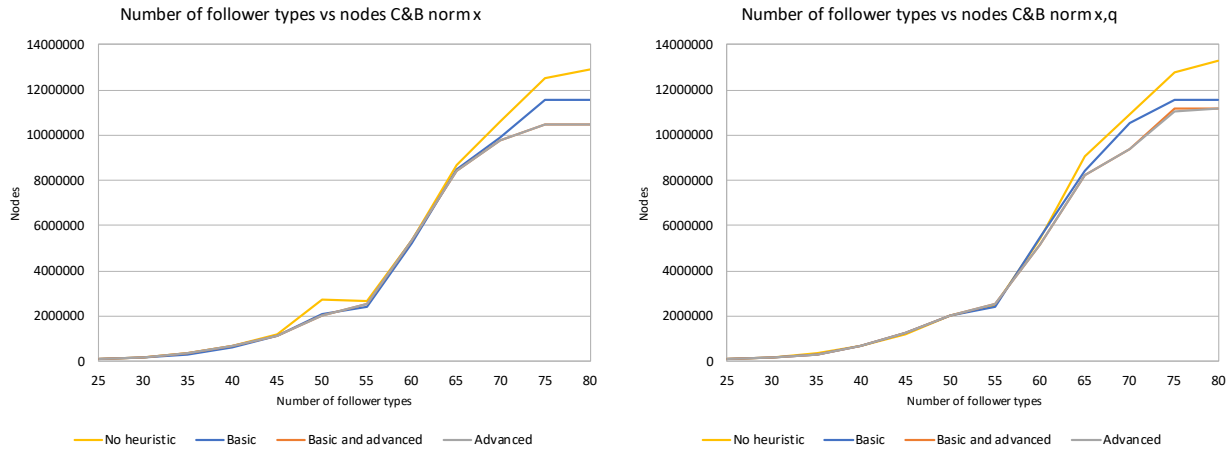

Figure 7.4: Effect of LB heuristics on average nodes explored for general instances A.

We can see that despite of the highest number of nodes explored, no LB heuristic performs better in average time for both algorithms. With this, we conclude that the best configuration is no LB heuristic for both algorithms.

Impact of upper bound enhancing heuristics Table (7.6) shows the average results with different UB configurations for set of general instances A. We can see that, for both algorithms, no UB heuristic produces the best average solution time. We can see that the best average UB improvement is achieved by the second parameter for both algorithms, which is consistent with the average time. The UB heuristic with parameter 0.01-0.25 explores the less number of average nodes for both algorithms. No UB heuristic explores the highest number of average nodes, more than 200% of the smallest.

Configuration	(C&B$_x$norm)	(C&B$_{x,q}$norm)
Average time		
No UB heuristic	**3401**	**3370**
0.01-0.1	3990	4015
0.01-0.25	3744	3493
0.9-0.99	3940	3874
Average UB improvement		
0.01-0.1	62.84%	63.97%
0.01-0.25	**67.76%**	**69.06%**
0.9-0.99	62.56%	63.58%
Average nodes		
No UB heuristic	4828435	4870837
0.01-0.1	3046069	2870734
0.01-0.25	**1881274**	**1757594**
0.9-0.99	3091600	2971586

Table 7.6: Algorithms average time, average UB improvement and average nodes for set of general instances A comparing different parameters of UB heuristics.

Figure (7.5) shows the effect of upper bound heuristics on average solution time for each number of follower types. In this graph, we present average value of the five instances in each size. We can see clearly that the best performance is produced for both algorithms with no UB heuristics, for the most of instance sizes.

Figure (7.6) shows the heuristic effect on average upper bound improvement for each number of follower types. In this graph, we present average value of the five instances in each size. We can see that, for every instance size, the second UB parameter outperforms others in both algorithms. This difference is bigger in medium instances.

Figure (7.7) shows the effect of upper bound heuristics on average number of nodes explored for each number of follower types. In this graph, we present average value of the five instances in each size. Yellow line that represents no UB heuristic configurations is above along the whole graph, which means that this configuration explores the highest number of nodes for each instance size. In the other hand, parameter 0.01-0.25 is below along the whole graph, which means the opposite.

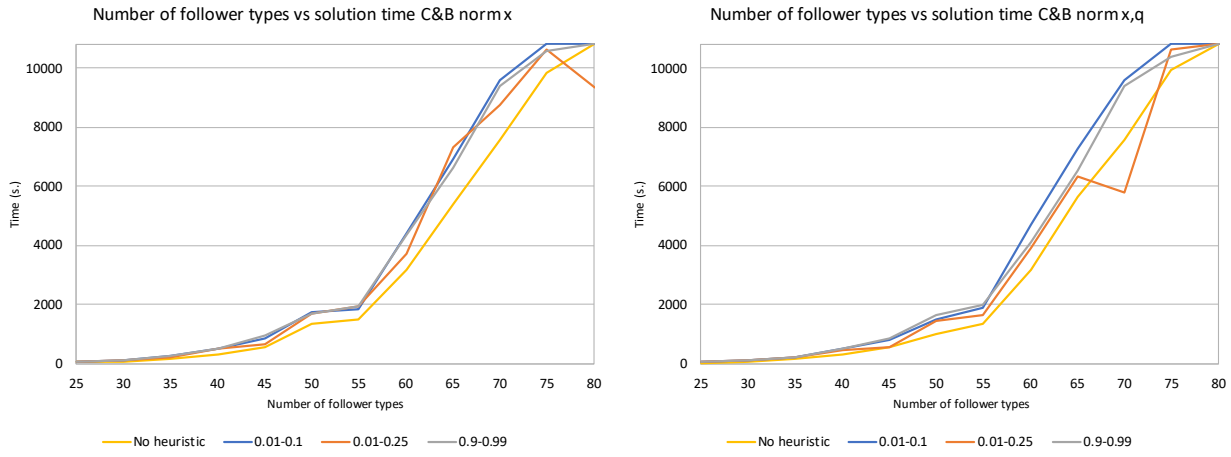

Figure 7.5: Effect of UB heuristics on average solution time for general instances A.

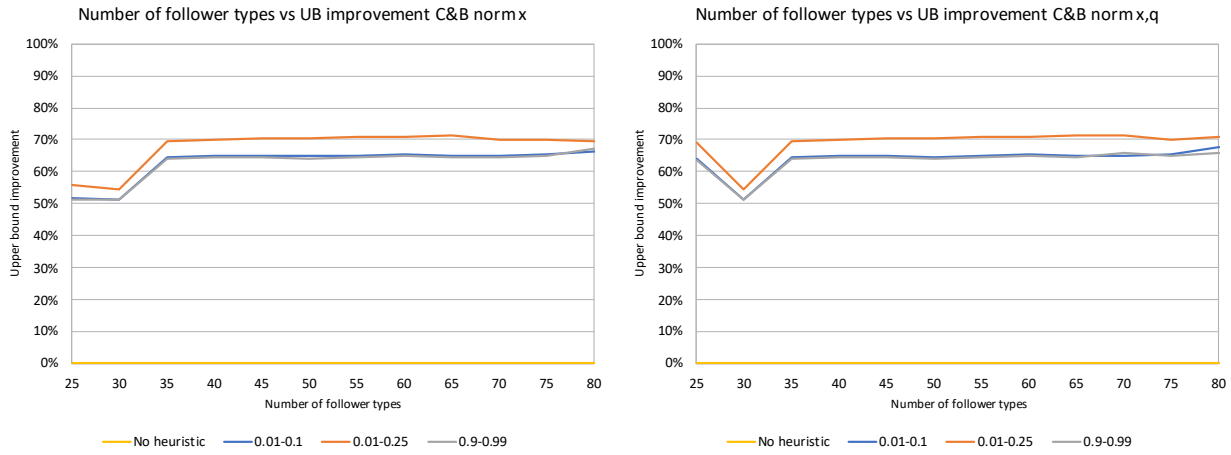

Figure 7.6: Effect of UB heuristics on average upper bound improvement for general instances A.

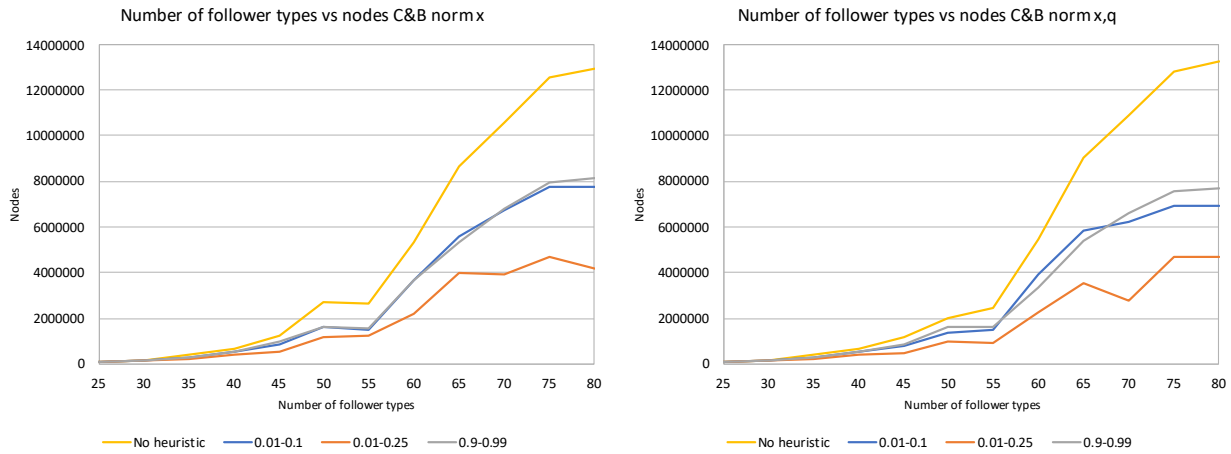

Figure 7.7: Effect of UB heuristics on average nodes explored for general instances A.

It is interesting to observe that the upper bound heuristics improve the UB and the nodes explored, but it can't be observed in the average solution time. It could be explained by the heuristic execution time, which is bigger than the time saved exploring less nodes.

With this analysis, we conclude that the upper bound heuristics don't improve the algorithms performance.

Best configuration Table (7.7) shows the best heuristic configuration chosen for the algorithms for this set of instances. This configuration is used in §7.5 to compare our algorithms with [19].

Algorithm	Stabilization	LB heuristics	UB heuristics
(C&B$_x$norm)	1	No LB heuristic	No UB heuristic
(C&B$_{x,q}$norm)	1	No LB heuristic	No UB heuristic

Table 7.7: Best configuration set of general instances A.

Instances B

Table (7.8) shows the performance of four algorithms with no heuristic configurations. The whole set of 30 instances is considered for this.

Algorithm	Av. time	% solve
(B&C$_q$)	7425	47%
(B&C$_q$norm)	6127	50%
(C&B$_x$norm)	**4930**	**70%**
(C&B$_{x,q}$norm)	**4823**	**70%**

Table 7.8: Algorithms performance for set of general instances B without heuristic configurations.

We can see that both cut & branch algorithms outperform branch & cut algorithms, in average time and percentage of solved instances. For that reason, we decide to try heuristic configurations only in these ones.

For trying the different heuristic configurations, we select a subset of instances: $I = J = \{5, 6, 7, 8\}$, $K = \{23\}$.

Impact of cut loop stabilization Table (7.9) shows average results for set of general instances B. $\lambda = 0.2$ means aggressive stabilization and $\lambda = 1$ means no stabilization. For this set of general instances, algorithm (C&B$_x$norm) performs a better average time using $\lambda = 0.2$, and (C&B$_{x,q}$norm) using $\lambda = 0.5$. We can see that $\lambda = 1$ performs faster relaxation time for both algorithms. For the first algorithm, we can see that there is not a big difference in number of nodes explored. If we observe, $\lambda = 1$ has a faster relaxation time than $\lambda = 0.2$ just for three hundredths of a second, but $\lambda = 0.2$ explores 50.000 less nodes, which explains the average time performance. In the second algorithm, $\lambda = 0.5$ takes 3 additional seconds

for solving the relaxation than $\lambda = 1$, but explores 500.000 less nodes. Apparently, it has an impact bigger than the relaxation time difference.

λ	(C&B$_x$norm)	(C&B$_{x,q}$norm)
Average time		
0.2	**1985**	1919
0.5	1997	**1711**
0.7	2001	1883
1	2003	1856
Average LP time		
0.2	13.72	8.39
0.5	14.22	7.58
0.7	18.38	6.93
1	**13.69**	**4.73**
Average nodes		
0.2	5234638	5135372
0.5	**5204186**	**4600292**
0.7	**5204186**	5031151
1	5283496	5072613

Table 7.9: Algorithms average time, average LP time and average nodes for set of general instances B comparing different stabilization parameters.

The analysis of figures here is similar to the set of instances A. For that reason, in this section we just summarize the results.

Figure (7.8) shows the percentage of problems solved as time goes by.

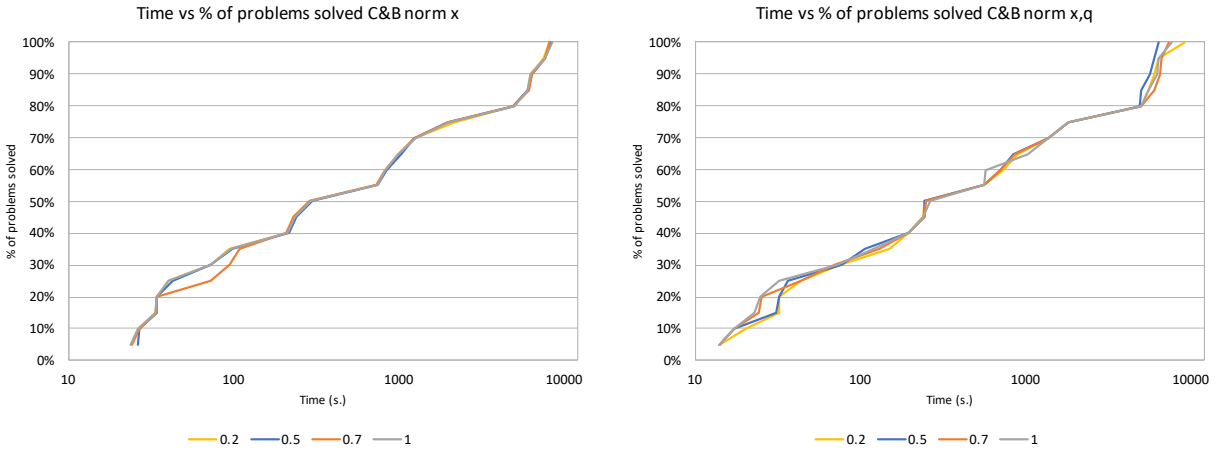

Figure 7.8: Effect of stabilization on solution time for general instances B.

We can conclude that the best stabilization parameters are $\lambda = 0.2$ and $\lambda = 0.5$, for (C&B$_x$norm) and (C&B$_{x,q}$norm), respectively.

Impact of lower bound enhancing heuristics Table (7.10) shows the average results for different heuristic configurations for set of general instances B. We can see that no LB heuristic has the very best average time performance in both algorithms. The time is 20% and 25% smaller than its nearest competitor for first and second algorithm. The relaxation is solved faster with no LB heuristic and basic LB heuristic for first and second algorithm, respectively. Basic LB heuristic explores the lowest number of nodes for both algorithms.

Configuration	(C&B$_x$norm)	(C&B$_{x,q}$norm)
Average time		
No LB heuristic	**1986**	**1711**
Basic LB heuristic	2750	2576
Basic and advanced LB heuristic	2557	2626
Advanced LB heuristic	2794	2595
Average LP time		
No LB heuristic	**13.72**	7.58
Basic LB heuristic	14.47	**6.68**
Basic and advanced LB heuristic	13.87	7.4
Advanced LB heuristic	14.4	7.36
Average nodes		
No LB heuristic	5234638	4600292
Basic LB heuristic	**4292678**	**4097152**
Basic and advanced LB heuristic	4752631	4353817
Advanced LB heuristic	4752631	4353817

Table 7.10: Algorithms average time, average LP time and average nodes for set of general instances B comparing different parameters of LB heuristics.

Figure (7.9), (7.10) and (7.11) show the effect of lower bound heuristics on average solution time, average relaxation time and average nodes explored for different number of pure strategies.

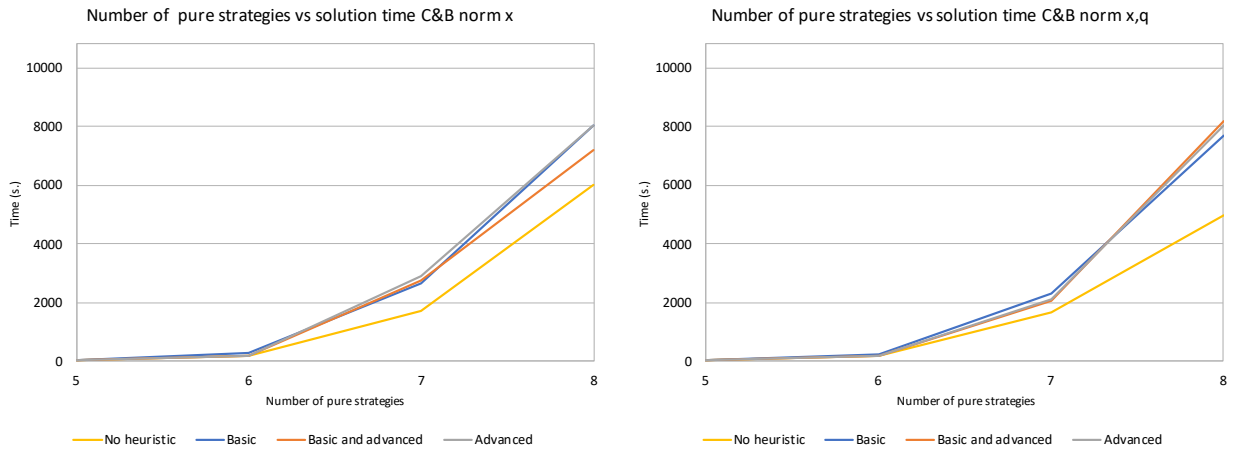

Figure 7.9: Effect of LB heuristics on average solution time for general instances B.

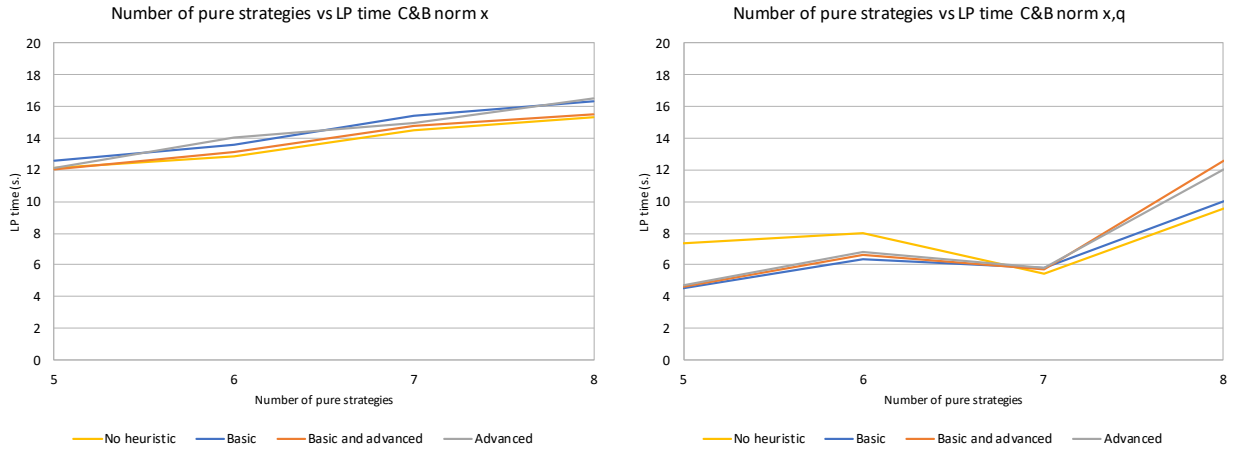

Figure 7.10: Effect of LB heuristics on average relaxation time for general instances B.

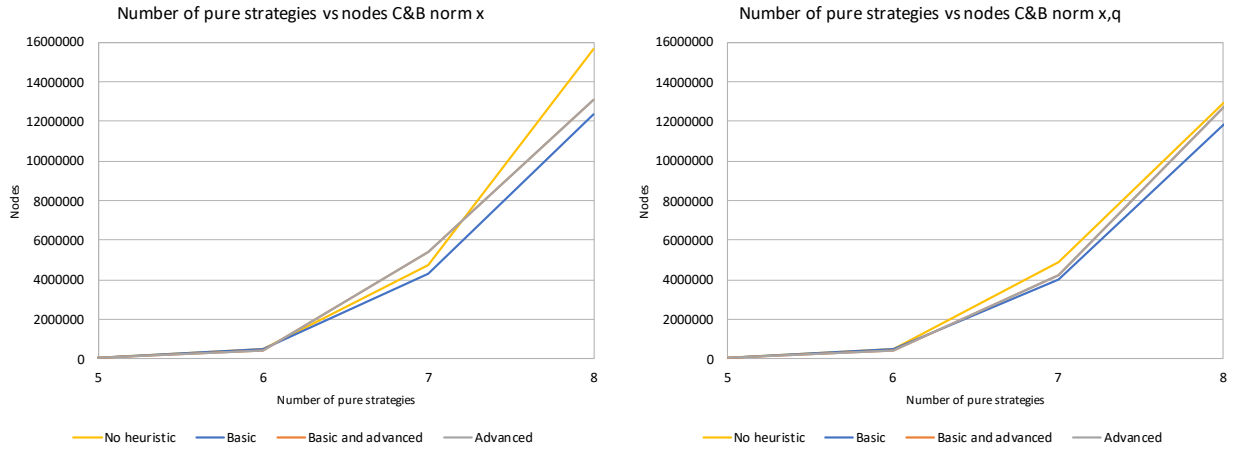

Figure 7.11: Effect of LB heuristics on average nodes explored for general instances B.

For the first algorithm, we can see that no LB heuristic has the best relaxation time and the worst number of nodes explored, but the general performance is better than the other configurations. In the second algorithm, no LB heuristic has the worst performance in average relaxation time and nodes explored, but better average time. We can't explain the internal process that occurs to get this, but we can conclude that that LB heuristics do not improve the performance of any algorithm.

Impact of upper bound enhancing heuristics Table (7.11) shows the average results with different UB configurations for set of general instances B. We can see that no UB heuristic outperforms the other configurations in average solution time for both algorithms. The second parameter 0.01-0.025 outperforms others in average UB improvement for both algorithms, for approximately 5%. We can see clearly that the UB heuristic with parameter 0.01-0.25 explores the less number of average nodes for both algorithms. No UB heuristic

explores the highest number of average nodes, more than 200% of the smallest. These results are similar than set of general instances A.

Configuration	(C&B$_x$norm)	(C&B$_{x,q}$norm)
Average time		
No UB heuristic	**1986**	**1711**
0.01-0.1	2152	2129
0.01-0.25	2098	2100
0.9-0.99	2176	1973
Average UB improvement		
0.01-0.1	64.67%	64.68%
0.01-0.25	**69.11%**	**69.04%**
0.9-0.99	63.86%	63.87%
Average nodes		
No UB heuristic	5234638	4600293
0.01-0.1	2794384	2599122
0.01-0.25	**2239967**	**2209362**
0.9-0.99	2833451	2756206

Table 7.11: Algorithms average time, average UB improvement and average nodes for set of general instances B comparing different parameters of UB heuristics.

Figure (7.12), (7.13) and (7.14) show the effect of upper bound heuristics in average solution, average upper bound improvement and average nodes explored for each number of pure strategies.

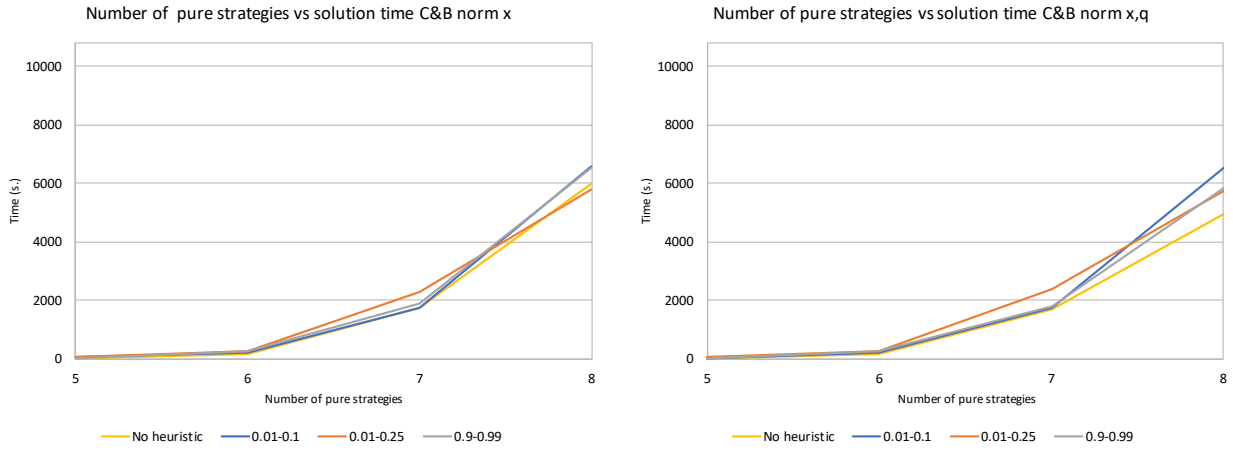

Figure 7.12: Effect of UB heuristics on average solution time for general instances B.

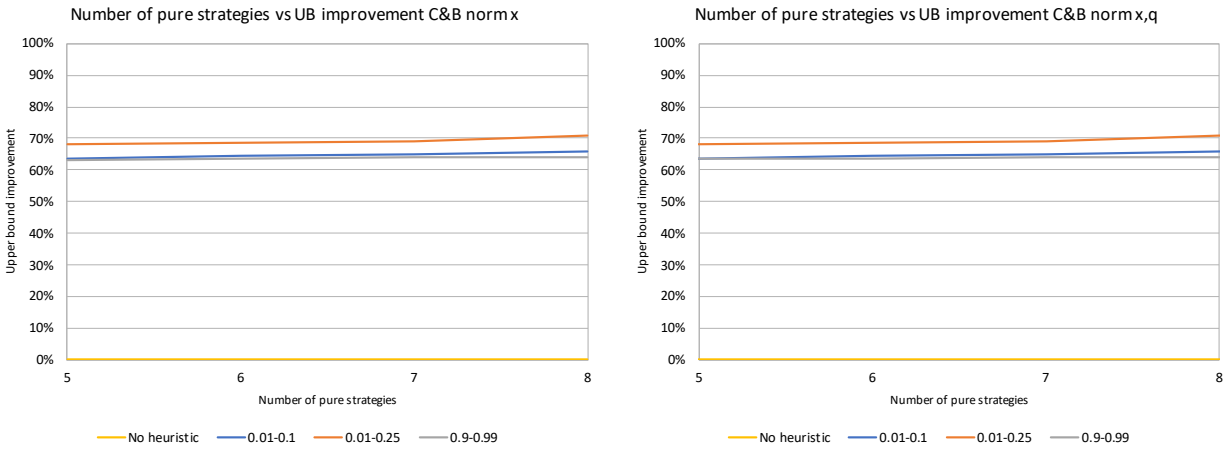

Figure 7.13: Effect of UB heuristics on average upper bound improvement for general instances B.

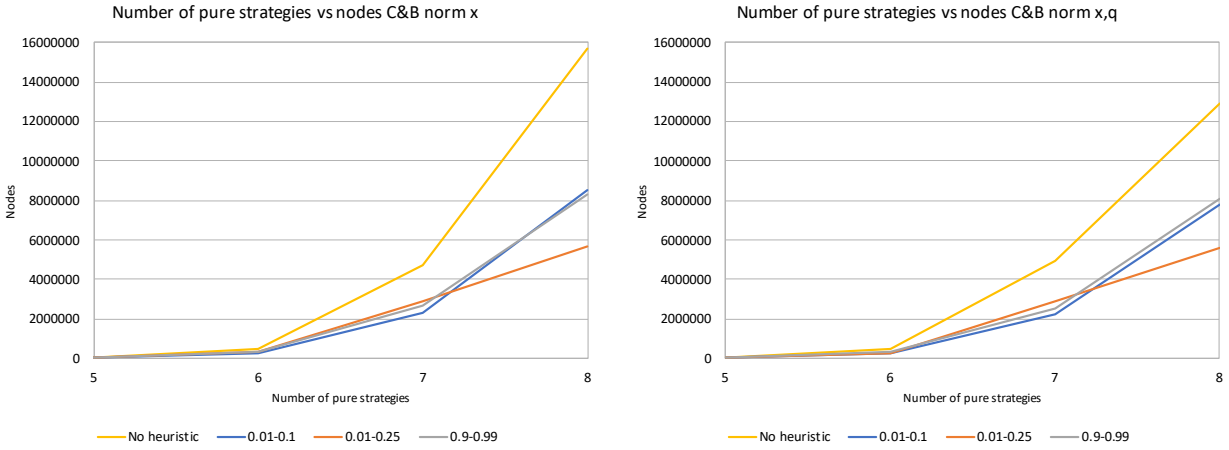

Figure 7.14: Effect of UB heuristics on average nodes explored for general instances B.

It is interesting to observe that the upper bound heuristics improve the UB and the nodes explored, but it can't be observed in the average solution time. It could be explained by the heuristic execution time, which is bigger than the time saved exploring less nodes.

Best configuration Table (7.12) shows the best heuristic configuration chosen for the algorithms for this set of instances. This configuration is used in §7.5 to compare our algorithms with [19].

Algorithm	Stabilization	LB heuristics	UB heuristics
(C&B$_x$norm)	0.2	No LB heuristic	No UB heuristic
(C&B$_{x,q}$norm)	0.5	No LB heuristic	No UB heuristic

Table 7.12: Best configuration set of general instances B.

7.4.2 Security Stackelberg games

In this section, we compare four different algorithms: branch & cut with q in master problem, normalized branch & cut with q in master problem, normalized cut & branch with c in master problem and normalized cut & branch with c and q in master problem. Complementary set of algorithms is compared in [19].

We separate experiments by set of instances, in order to identify the best algorithm for each type of them separately.

Instances A

Table (7.13) shows the performance of four algorithms with no heuristic configurations. The whole set of 55 instances is considered for this.

Algorithm	Av. time	% solve
(B&C$_q$)	-	0%
(B&C$_q$norm)	-	0%
(C&B$_c$norm)	**3316**	**98%**
(C&B$_{c,q}$norm)	**1570**	**100%**

Table 7.13: Algorithms performance for set of security instances A without heuristic configurations.

First and second algorithms didn't solve any instance, and the other two solved almost all instances. For that reason, we continue trying heuristic configurations in these two.

For trying the different heuristic configurations, we select a subset of instances: $J = \{25\}$, $K = \{45, 50, \ldots, 80\}$.

Impact of cut loop stabilization Table (7.14) shows the average results of both algorithms with different stabilization parameters. We can see that average solution time is better using parameter $\lambda = 1$ for both normalized cut & branch algorithms. The first algorithm solves the relaxation faster when it is not stabilizing, but the difference between parameters is very small, less than 2 second with the farthest. In the second algorithm there is no difference between parameters. For the first algorithm, $\lambda = 1$ explores 4.0000 more nodes than its hardest competitor, however it does this with the fastest average solution time. The second algorithm doesn't show any difference of average nodes explored for different parameters. So, stabilization doesn't have any effect in the average performance.

Figure (7.15) shows the percentage of problems solved as time goes by. Time limit is 10.800 seconds (3 hours). For the first algorithm, we can't see clearly a grey line separated from others, except in the slower algorithms, where is above others. In the second algorithm, is impossible distinguish the lines, which is consistent with the tight difference of average solution times in table (7.14).

λ	(C&B$_c$norm)	(C&B$_{c,q}$norm)
Average time		
0.2	1750	870
0.5	1806	869
0.7	1734	870
1	**1732**	**868**
Average LP time		
0.2	27.88	1.25
0.5	27.94	1.25
0.7	27.11	1.25
1	**26.5**	1.25
Average nodes		
0.2	2146771	1206877
0.5	2146771	1206877
0.7	**2140763**	1206877
1	2144672	1206877

Table 7.14: Algorithms average time, average LP time and average nodes for set of security instances A comparing different stabilization parameters.

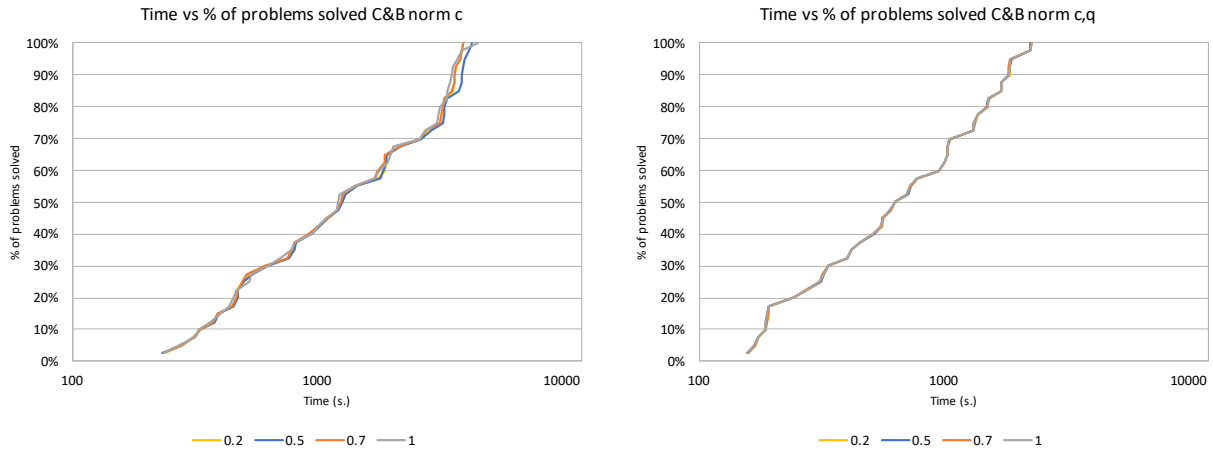

Figure 7.15: Effect of stabilization on solution time for security instances A.

With this analysis, we can conclude that $\lambda = 1$ is the best stabilization parameter for this set of security instances.

Impact of lower bound enhancing heuristics Table (7.15) shows the average results for different LB heuristic configurations. We can see that no LB heuristic performs better in average time for both algorithms. No LB heuristic solves the relaxation in an average time better than other configurations for both algorithms. We can see that basic LB heuristic explores the lowest number of average nodes in the branching tree for both algorithms. Is interesting to see that the differences are less than 15% between configurations in both algorithms.

Configuration	(C&B$_c$norm)	(C&B$_{c,q}$norm)
Average time		
No LB heuristic	**1732**	**868**
Basic LB heuristic	2260	980
Basic and advanced LB heuristic	2348	981
Advanced LB heuristic	2280	984
Average LP time		
No LB heuristic	**26.5**	**1.25**
Basic LB heuristic	28	3.61
Basic and advanced LB heuristic	28.35	1.29
Advanced LB heuristic	27.78	1.28
Average nodes		
No LB heuristic	2144672	1206877
Basic LB heuristic	**2109396**	**1039362**
Basic and advanced LB heuristic	2124009	1057649
Advanced LB heuristic	2124009	1057649

Table 7.15: Algorithms average time, average LP time and average nodes for set of security instances A comparing different parameters of LB heuristics.

Figure (7.16) shows the effect of lower bound heuristics on average solution time for different number of attacker types. In this graph, we present average value of the five instances in each size. We can see that for every instance size, no LB heuristic performs better than others, for both algorithms, although in the first algorithm the difference is pretty much clearer.

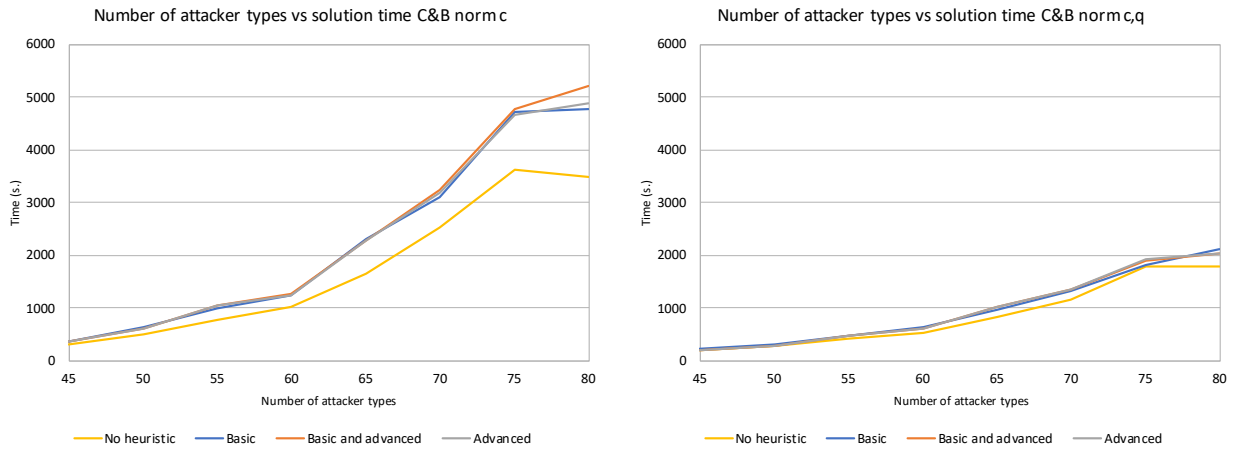

Figure 7.16: Effect of LB heuristics on average solution time for security instances A.

Figure (7.17) shows the effect of lower bound heuristics on average relaxation time for different number of attacker types. In this graph, we present average value of the five instances in each size. For the first algorithm, we see that no LB heuristic performs better. For the

second algorithm, we see that in almost every instance size the relaxation time is almost the same, except in the instances of 45 and 50 attacker types, where is pretty higher.

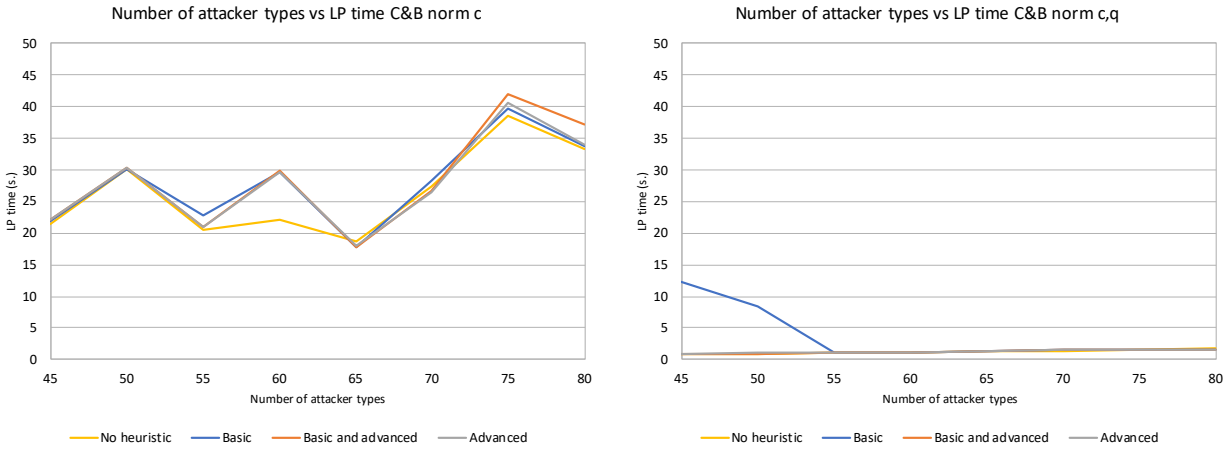

Figure 7.17: Effect of LB heuristics on average relaxation time for security instances A.

Figure (7.18) shows the effect of lower bound heuristics on the average number of nodes explored for different number of attacker types. In this graph, we present average value of the five instances in each size. For the first algorithm, we see that basic LB heuristic performs better in most of the graph, and no LB heuristic is the worst. For the second algorithm, the results are similar.

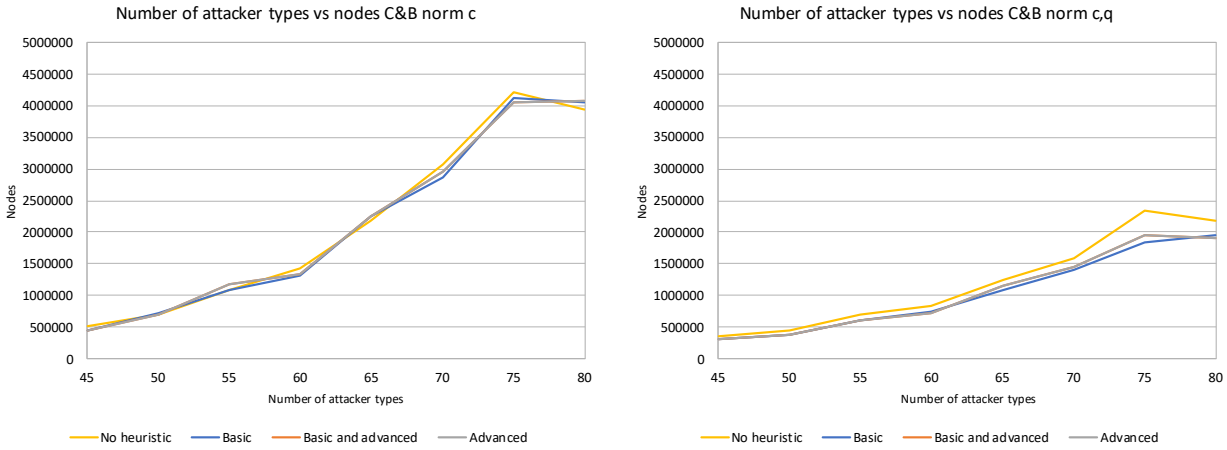

Figure 7.18: Effect of LB heuristics on average nodes explored for security instances A.

In both algorithms, no LB heuristic has the fastest relaxation time, but the highest number of nodes. We see that the time of processing those extra nodes is not enough to make it slower than others. For that reason, we conclude that LB heuristics don't improve the algorithms performance for this set of instances.

Impact of upper bound enhancing heuristics Table (7.16) shows the average results with different UB configurations for set of security instances A. Both algorithms have better performances in avergae time when they are not using UB heuristics. The best average UB improvement is achieved for the second parameter for both algorithms. We can see The UB heuristic with parameter 0.9-0.99 explored the less number of average nodes for both algorithms and no UB heuristic explores the highest number of average nodes.

Configuration	(C&B$_c$norm)	(C&B$_{c,q}$norm)
Average time		
No UB heuristic	**1732**	**868**
0.01-0.1	3436	1507
0.01-0.25	3477	1555
0.9-0.99	3122	1388
Average UB improvement		
0.01-0.1	72.02%	73.05%
0.01-0.25	**73.36%**	**74.35%**
0.9-0.99	71.74%	72.83%
Average nodes		
No UB heuristic	2144673	1206877
0.01-0.1	2014947	960204
0.01-0.25	1969874	991641
0.9-0.99	**1900627**	**946540**

Table 7.16: Algorithms average time, average UB improvement and average nodes for set of security instances A comparing different parameters of UB heuristics.

Figure (7.19) shows the effect of upper bound heuristics on average solution time for each number of attacker types. In this graph, we present average value of the five instances in each size. We can see clearly that the best performance is produced for both algorithms with no UB heuristics, for every instance size.

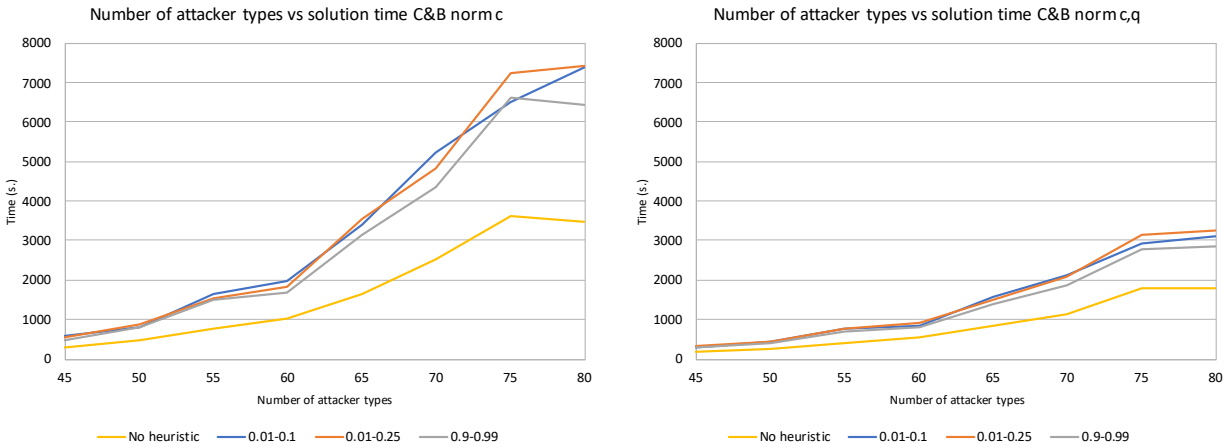

Figure 7.19: Effect of UB heuristics on average solution time for security instances A.

Figure (7.20) shows the heuristic effect on average upper bound improvement for each number of attacker types. In this graph, we present average value of the five instances in each size. We can see with the orange line that, for every instance size, the second UB parameter outperforms others in both algorithms.

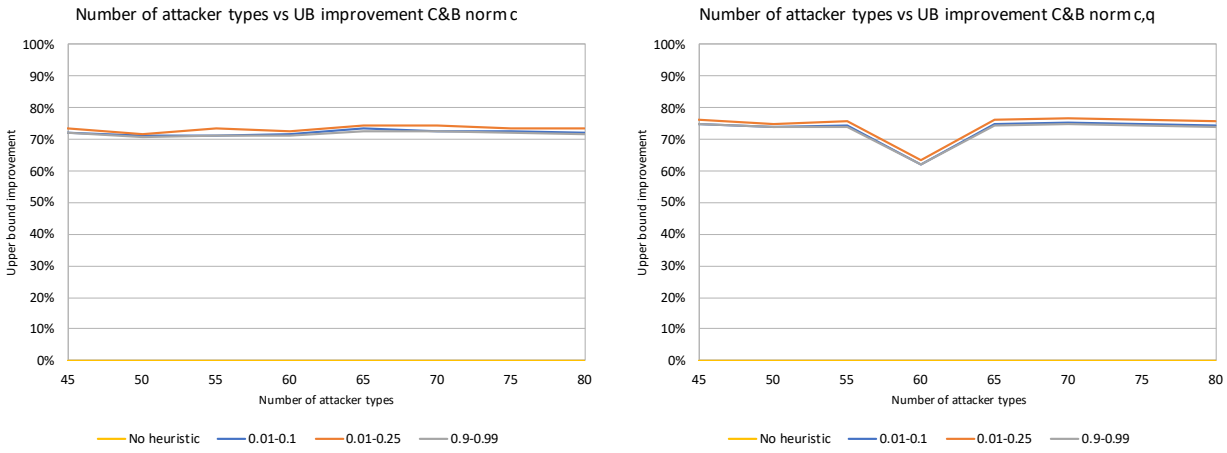

Figure 7.20: Effect of UB heuristics on average upper bound improvement for security instances A.

Figure (7.21) shows the effect of upper bound heuristics on average number of nodes explored for each number of attacker types. In this graph, we present average value of the five instances in each size. Yellow line that represents no UB heuristic is above in most of the graph (in both algorithms), which means that for separate instances no UB heuristics explores the highest number of nodes too. Conversely, grey line is below in most of the graph, which means that the parameter 0.9-0.99 explores the smallest number of nodes in each instance size.

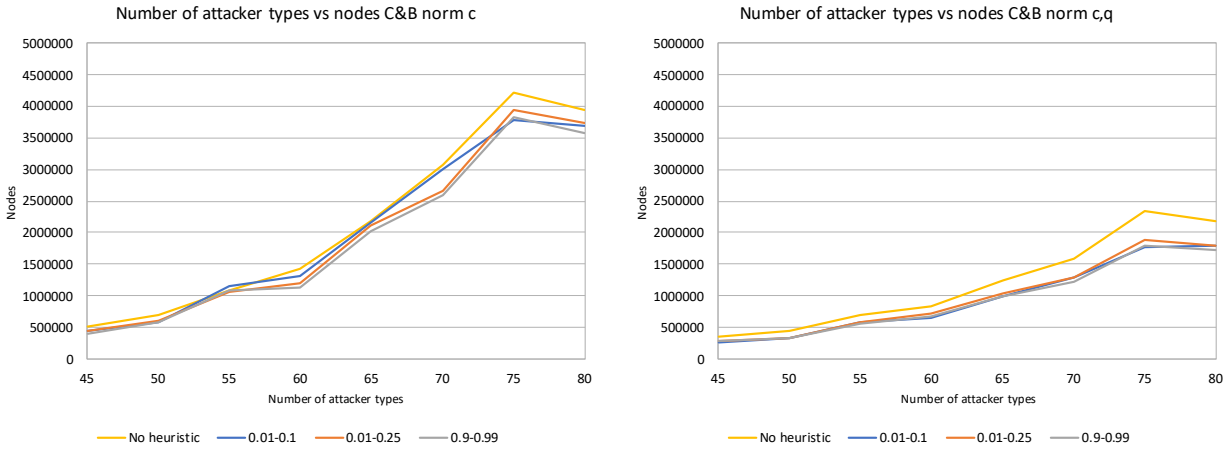

Figure 7.21: Effect of UB heuristics on average nodes explored for security instances A.

Upper bound heuristic configurations produce big improvements to the upper bound after solving the root node, which leads to explore less nodes. But time saved in solving less nodes doesn't make up the time of executing the heuristic algorithm. For that reason, average time is better in absence of UB heuristics.

Best configuration Table (7.17) shows the best heuristic configuration chosen for the algorithms for this set of instances. This configuration is used in §7.5 to compare our algorithms with [19].

Algorithm	Stabilization	LB heuristics	UB heuristics
(C&B$_c$norm)	1	No LB heuristic	No UB heuristic
(C&B$_{c,q}$norm)	1	No LB heuristic	No UB heuristic

Table 7.17: Best configuration set of security instances A.

Instances B

Table (7.18) shows the performance of four algorithms with no heuristic configurations. The whole set of 20 instances is considered for this.

Algorithm	Av. time	% solve
(B&C$_q$)	-	0%
(B&C$_q$norm)	-	0%
(C&B$_c$norm)	**8458**	**35%**
(C&B$_{c,q}$norm)	**8304**	**40%**

Table 7.18: Algorithms performance for set of security instances B without heuristic configurations.

We can see that both cut & branch algorithms outperform branch & cut algorithms, in average time and percentage of solved instances. For that reason, we decide to try heuristic configurations only in these ones.

For trying the different heuristic configurations, we select a subset of instances: $I = J = \{8, 10, 12\}$, $K = \{25\}$.

Impact of cut loop stabilization Table (7.19) shows average results for both algorithms with different stabilization parameters. We can see that the first algorithm performs better in average time when using $\lambda = 1$, but with a difference of less than 11 seconds with all competitors. The second algorithms performs better when using $\lambda = 0.7$ but, as the first algorithm, presents a little difference with competitors. For the first algorithm, we can see that the average relaxation time is very similar for the set of parameters, except for $\lambda = 0.2$, where it takes 50% more time. The second algorithm presents similar performances for the set of parameters, except for $\lambda = 1$, where it takes a 50% more of time. For both algorithms, each stabilization parameter presents the same number of average nodes explored.

λ	(C&B$_c$norm)	(C&B$_{c,q}$norm)
Average time		
0.2	7693	7485
0.5	7693	7478
0.7	7684	**7469**
1	**7678**	7472
Average LP time		
0.2	3.05	1.16
0.5	2.47	**1.11**
0.7	**2.43**	1.22
1	2.47	1.94
Average nodes		
0.2	11331462	11661948
0.5	11331462	11661948
0.7	11331462	11661948
1	11331462	11661948

Table 7.19: Algorithms average time, average LP time and average nodes for set of security instances B comparing different stabilization parameters.

The analysis of figures here is similar to the set of instances A. For that reason, in this section we just summarize the results.

Figure (7.22) shows the percentage of problems solved as time goes by. We can see that lines can't be separated in both graphs, which is consistent with table (7.19), the difference in performance is not clear.

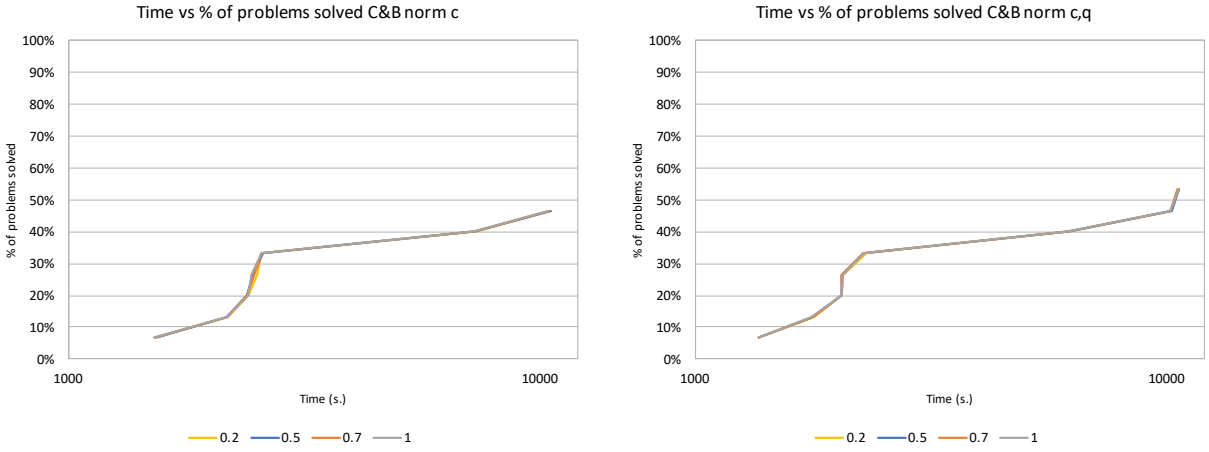

Figure 7.22: Effect of stabilization on solution time for security instances B.

In spite of the little differences in performance of parameters, we decided to configure the first algorithm with $\lambda = 1$, and the second algorithm with $\lambda = 0.7$.

Impact of lower bound enhancing heuristics Table (7.20) shows the average results for different LB heuristic configurations. No LB heuristic performs better than others in average solution time for both algorithms. No LB heuristic and basic LB heuristic perform better average relaxation time for first and second algorithm, respectively. For the first algorithm, we see that basic LB heuristic explores the lowest number of nodes. For the second algorithm, the lowest number of nodes is shared between basic and advanced LB heuristic, and advanced LB heuristic. In both algorithms, the differences are huge.

Configuration	(C&B$_c$norm)	(C&B$_{c,q}$norm)
Average time		
No LB heuristic	**7678**	**7469**
Basic LB heuristic	7866	7673
Basic and advanced LB heuristic	7914	9134
Advanced LB heuristic	7896	8980
Average LP time		
No LB heuristic	**2.47**	1.11
Basic LB heuristic	2.64	**1.1**
Basic and advanced LB heuristic	3.69	1.65
Advanced LB heuristic	2.68	1.23
Average nodes		
No LB heuristic	11331462	11661948
Basic LB heuristic	**6931852**	6819968
Basic and advanced LB heuristic	7363507	**3149237**
Advanced LB heuristic	7363507	**3149237**

Table 7.20: Algorithms average time, average LP time and average nodes for set of security instances B comparing different parameters of LB heuristics.

Figure (7.23),(7.24) and (7.25) show the effect of lower bound heuristics in average solution time, average relaxation time and average nodes explored for different number of targets.

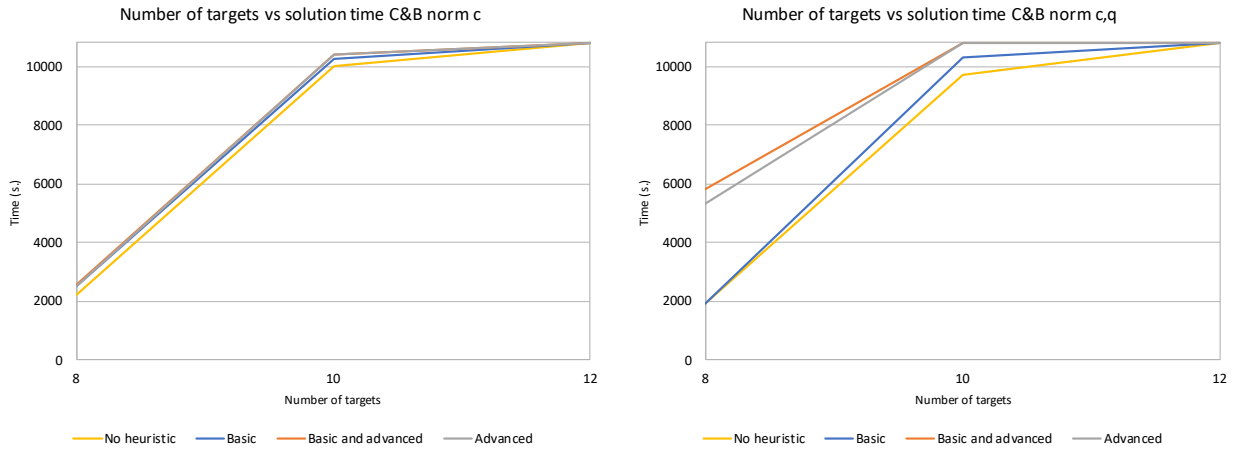

Figure 7.23: Effect of LB heuristics on average solution time for security instances B.

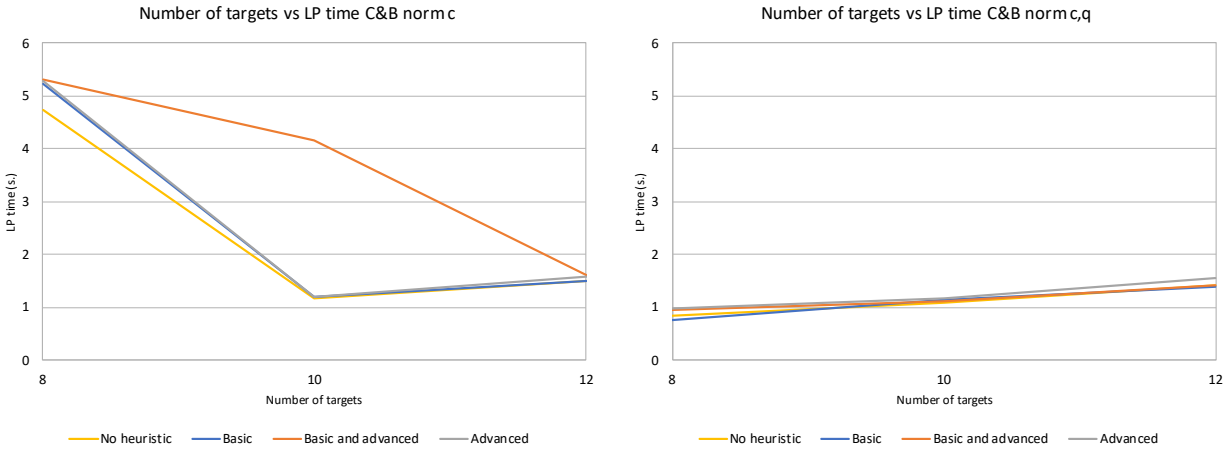

Figure 7.24: Effect of LB heuristics on average relaxation time for security instances B.

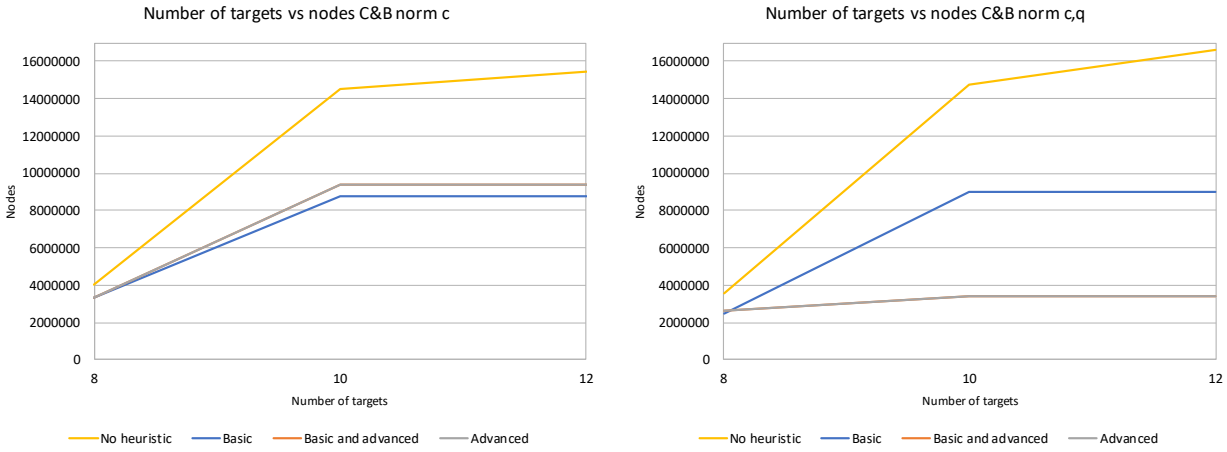

Figure 7.25: Effect of LB heuristics on average nodes explored for security instances B.

We can conclude that LB heuristic don't improve the performance of the algorithms.

Impact of upper bound enhancing heuristics Table (7.21) shows the average results with different UB configurations for set of security instances B. We can see that, for both algorithms, no UB heuristic performs better solution times. We can see that the second parameter presents the best average UB improvement for both algorithms. For the first algorithm, we see that parameter 0.01-0.1 explores the smallest number of nodes. For the second algorithm, it occurs with parameter 0.9-0.99. The worst performance occurs with no UB heuristic for both algorithms.

Figure (7.26), (7.27) and (7.28) show the effect of upper bound heuristics in average solution, average upper bound improvement and average nodes explored for each number of targets.

Configuration	(C&B$_c$norm)	(C&B$_{c,q}$norm)
Average time		
No UB heuristic	**7678**	**7469**
0.01-0.1	8282	8154
0.01-0.25	9110	8760
0.9-0.99	8319	8140
Average UB improvement		
0.01-0.1	73.40%	73.35%
0.01-0.25	**75.84%**	**75.95%**
0.9-0.99	73.19%	72.80%
Average nodes		
No UB heuristic	11331462	11661948
0.01-0.1	**2344616**	2383662
0.01-0.25	4387286	3218063
0.9-0.99	2742852	**2304021**

Table 7.21: Algorithms average time, average LP time and average nodes for set of security instances B comparing different parameters of UB heuristics.

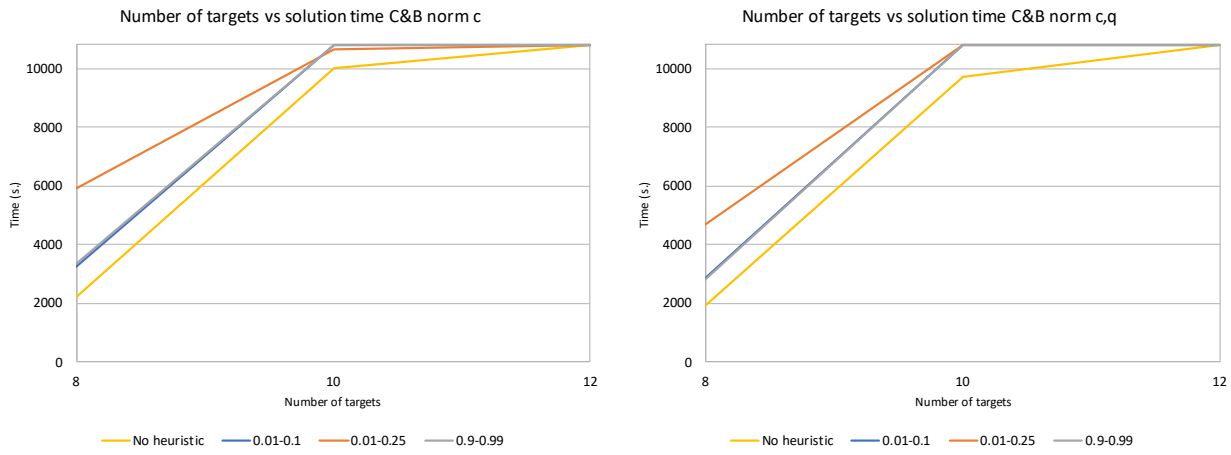

Figure 7.26: Effect of UB heuristics on average solution time for security instances B.

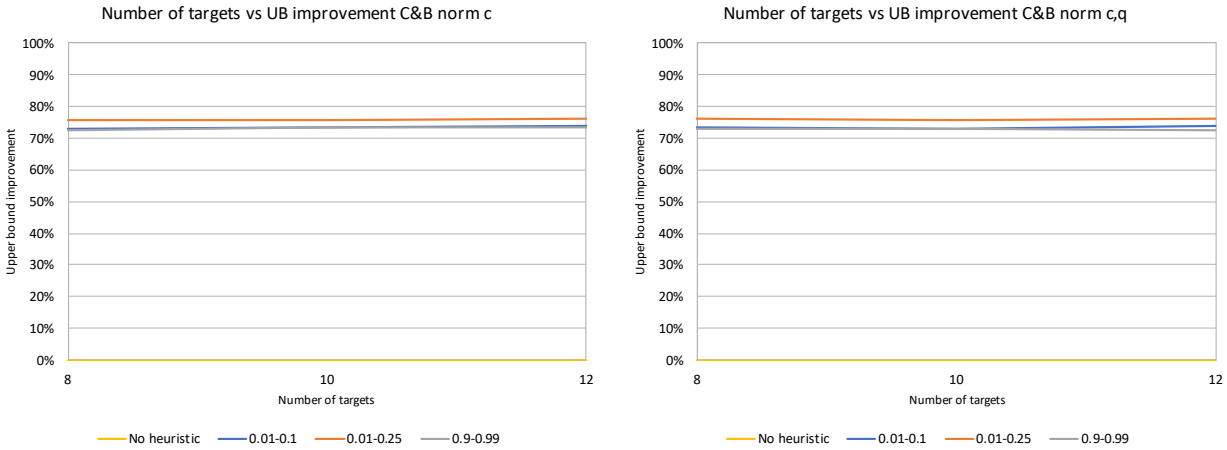

Figure 7.27: Effect of UB heuristics on average upper bound improvement for security instances B.

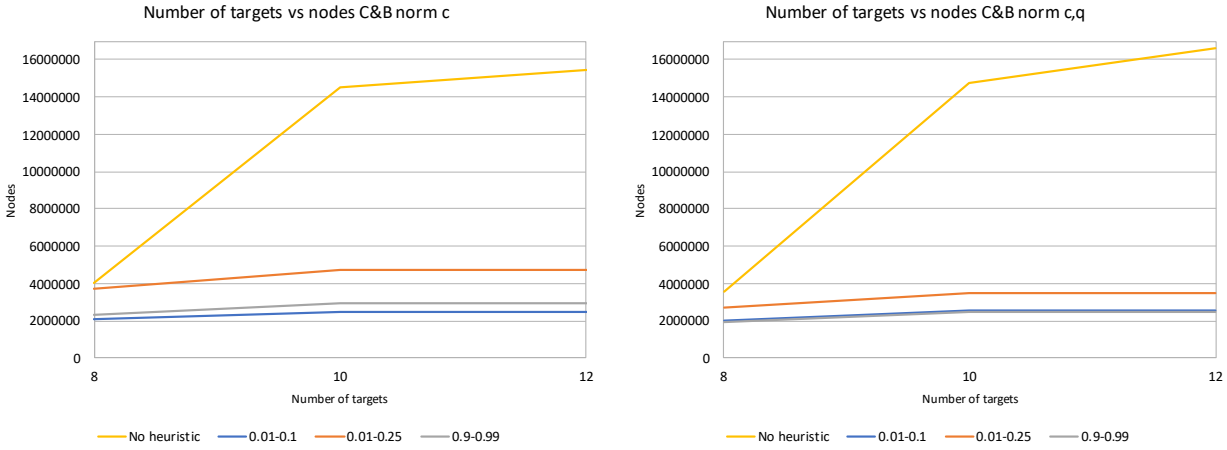

Figure 7.28: Effect of UB heuristics on average nodes explored for security instances B.

Upper bound heuristic configurations produce big improvements to the upper bound after solving the root node, which leads to explore less nodes. But time saved in solving less nodes doesn't make up the time of executing the heuristic algorithm. For that reason, average time is better in absence of UB heuristics.

Best configuration Table (7.22) shows the best heuristic configuration chosen for the algorithms for this set of instances. This configuration is used in §7.5 to compare our algorithms with [19].

Algorithm	Stabilization	LB heuristics	UB heuristics
(C&B$_c$norm)	1	No LB heuristic	No UB heuristic
(C&B$_{c,q}$norm)	0.7	No LB heuristic	No UB heuristic

Table 7.22: Best configuration set of security instances B.

7.5 Comparing algorithms

In this section, we compare our best algorithms chosen in §7.4 with those presented in [19].

7.5.1 General Stackelberg games

Instances A

We compare our best configured algorithms against the (C&B$_{x,q}$) algorithm, which was the best in [19] for this set of instances. Its best configuration is: 0.2 stabilization and no LB or UB heuristic.

Figure (7.29) shows the average solution time for the different algorithms, scaling up the number of follower types. We can see that (C&B$_{x,q}$) taken from [19] is the fastest solution method for every number of follower types.

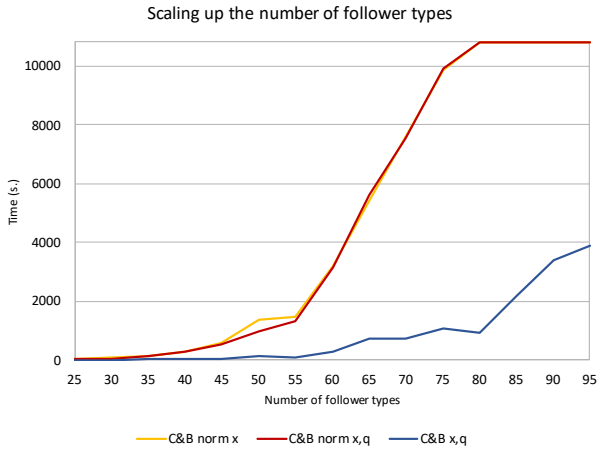

Figure 7.29: Average solution time scaling up the number of follower types.

Figure (7.30) presents performance profile graphs to compare solution time, relaxation time, solution time of the last linear problem of the root node relaxation and number of nodes explored.

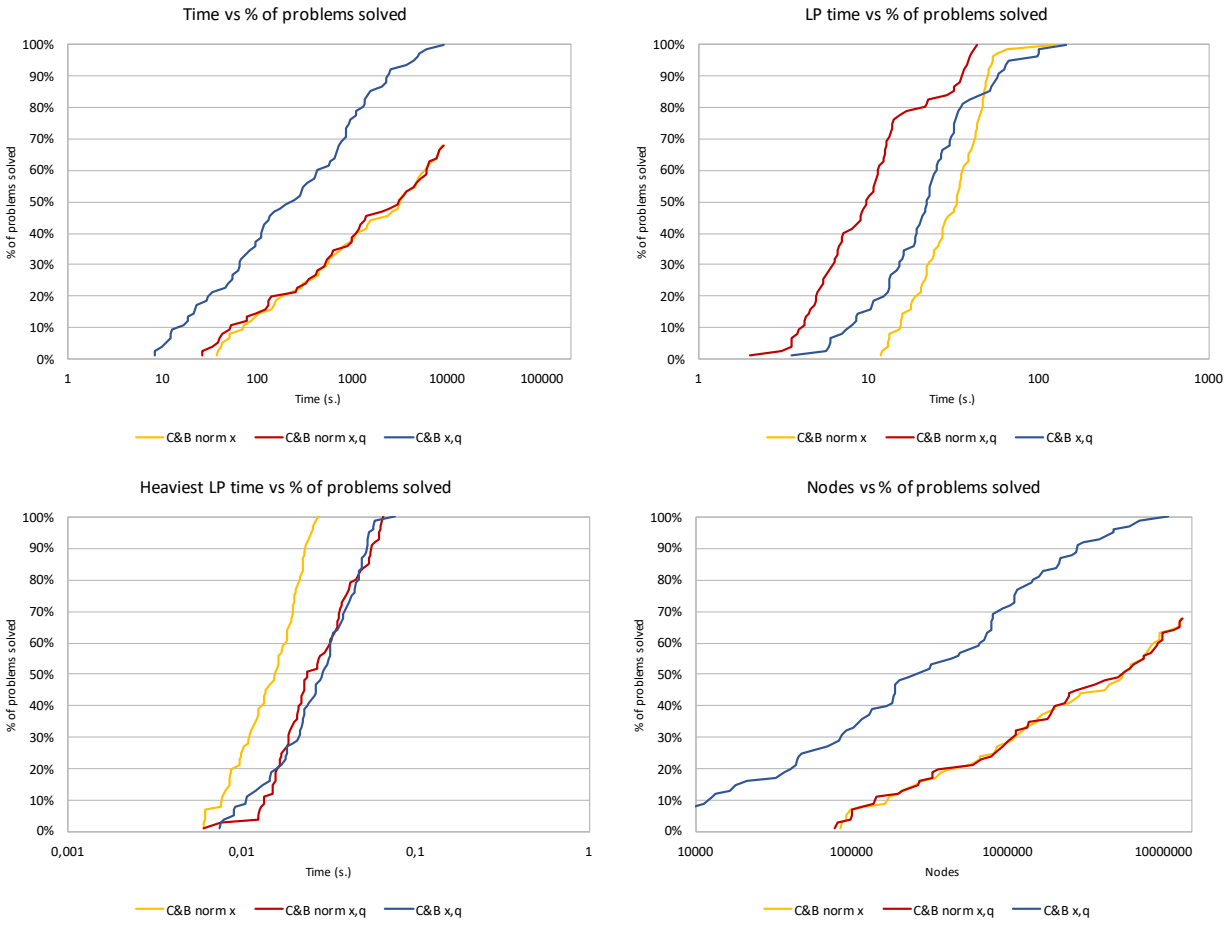

Figure 7.30: Performance profile graphs over general instances A.

We can see that (C&B$_{x,q}$) is the fastest algorithm, and (C&B$_x$norm) is the slowest. With regard to relaxation time, we observe that (C&B$_{x,q}$norm) has the fastest LP time. If we compare the solution time of the last linear problem of the root node, we see that (C&B$_x$norm) produces relaxations easier to solve than others. Finally, we can see that (C&B$_{x,q}$) explored the smallest number of nodes. With this information, we can say that the linear relaxations of (C&B$_{x,q}$) are better than for other algorithms, and this causes less nodes explored that make up for the slower relaxations, which leads to better solution times. The average relaxation objectives are drawn in figure (7.31), where the (C&B$_{x,q}$) relaxations of [19] are clearly better than our relaxations.

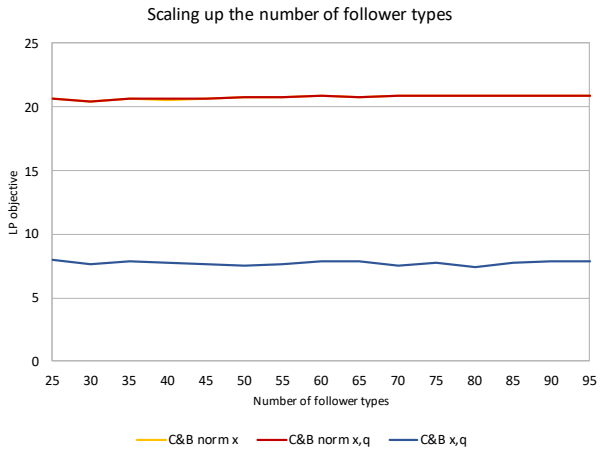

Figure 7.31: Average linear relaxation objective scaling up the number of follower types.

Instances B

We compare our best configured algorithms against (C&B$_x$) algorithm, which was the best in [19] for this set of instances. Its best configuration is: 0.5 stabilization and no LB or UB heuristic.

Figure (7.32) shows the average solution time for the different algorithms, scaling up the number of pure strategies. We can see that (C&B$_x$) taken from [19] is the fastest solution method for every number of pure strategies.

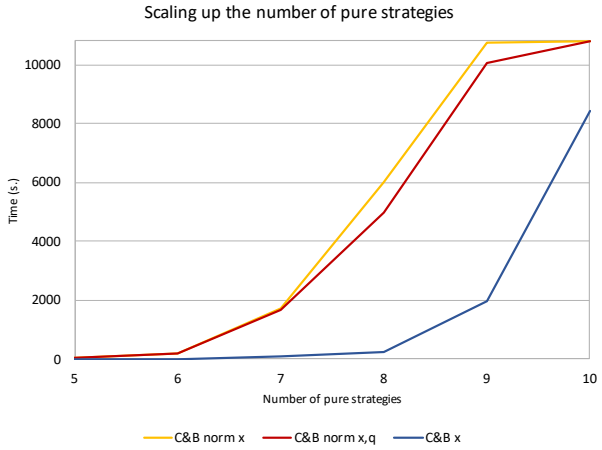

Figure 7.32: Average solution time scaling up the number of pure strategies.

Figure (7.33) presents performance profile graphs to compare solution time, LP solution time, solution time of the last linear problem of the root node relaxation and nodes explored.

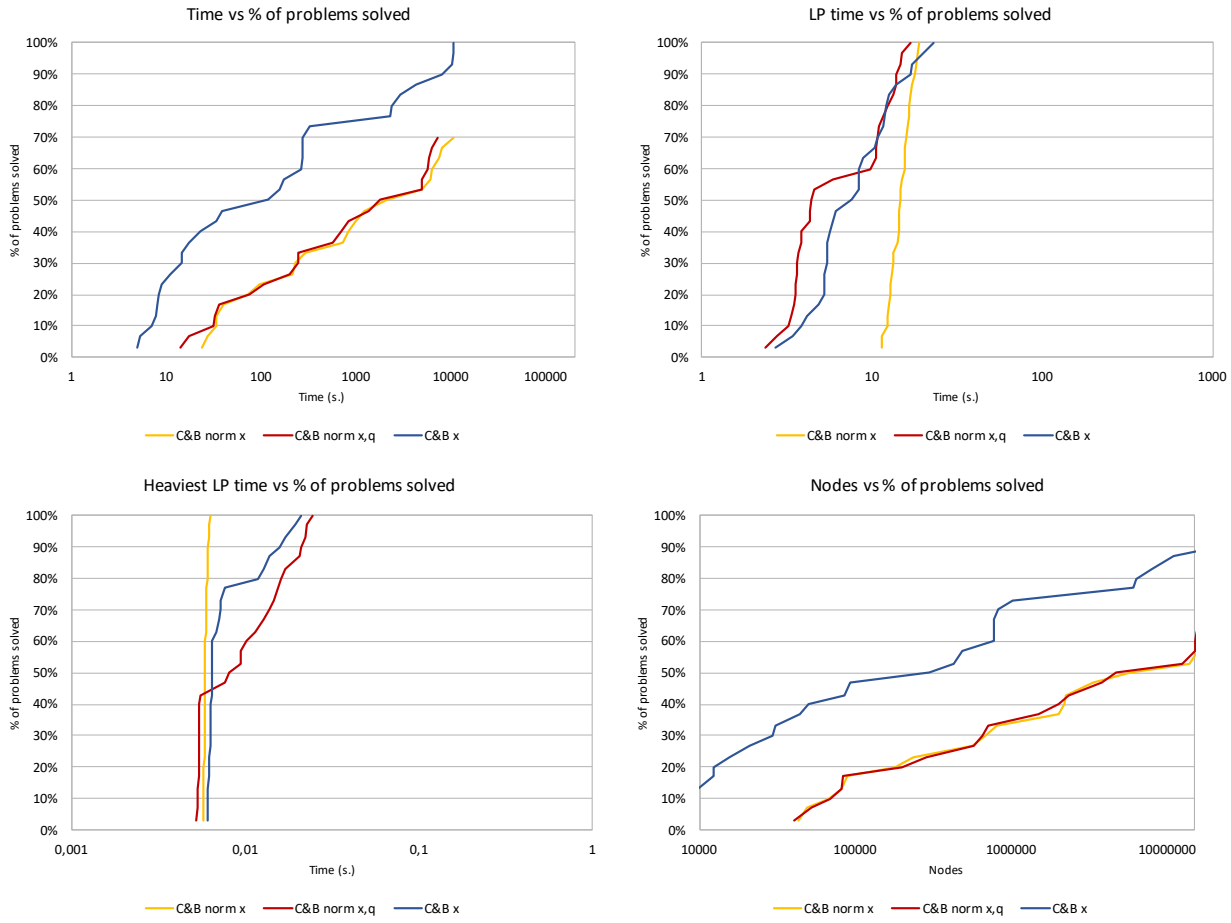

Figure 7.33: Performance profile graphs over general instances B.

We can see that (C&B$_x$) is the fastest algorithm, and (C&B$_x$norm) is the slowest, as in the set of general instances A. With regard to relaxation time, we observe that (C&B$_{x,q}$norm) has the fastest LP time in the most of instances, and (C&B$_x$norm) has the slowest. If we compare the solution time of the last linear problem of the root node, we see that (C&B$_x$norm) produces relaxations easier to solve than others in the most of instances. Finally, we can see that (C&B$_x$) explores the smallest number of nodes. With this information, we can say that the linear relaxations of (C&B$_x$) are better than for other algorithms, and this causes less nodes explored that make up for the slower relaxations, which leads to better solution times. This can be seen in figure (7.34), where the linear relaxations of (C&B$_x$) are much better than ours. These results are almost identical to the set of general instances A.

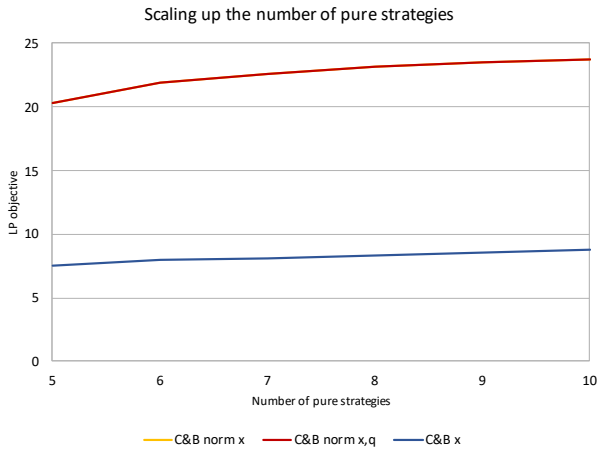

Figure 7.34: Average linear relaxation objective scaling up the number of pure strategies.

7.5.2 Security Stackelberg games

Instances A

We compare our best configured algorithms against (C&B$_{c,q}$) algorithm, which was the best in [19] for this set of instances. Its best configuration is: 0.7 stabilization, advanced LB heuristic and UB heuristic with parameter $0.01 - 0.25$.

Figure (7.35) shows the average solution time for the different algorithms, scaling up the number of attacker types. We can see that (C&B$_{c,q}$) taken from [19] is the fastest solution method for every number of attacker types.

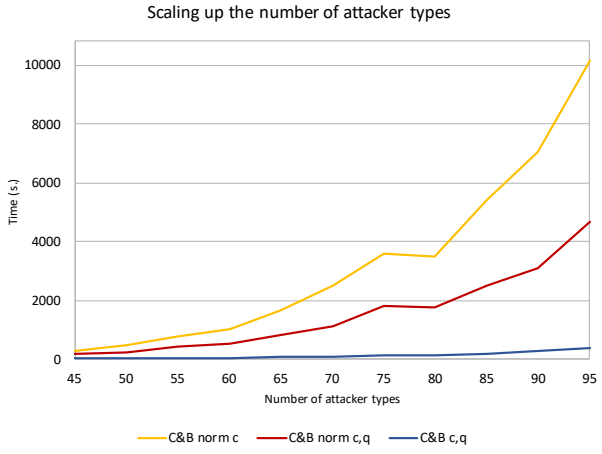

Figure 7.35: Average solution time scaling up the number of attacker types.

Figure (7.36) presents performance profile graphs to compare solution time, LP solution time, solution time of the last linear problem of the root node relaxation and nodes explored.

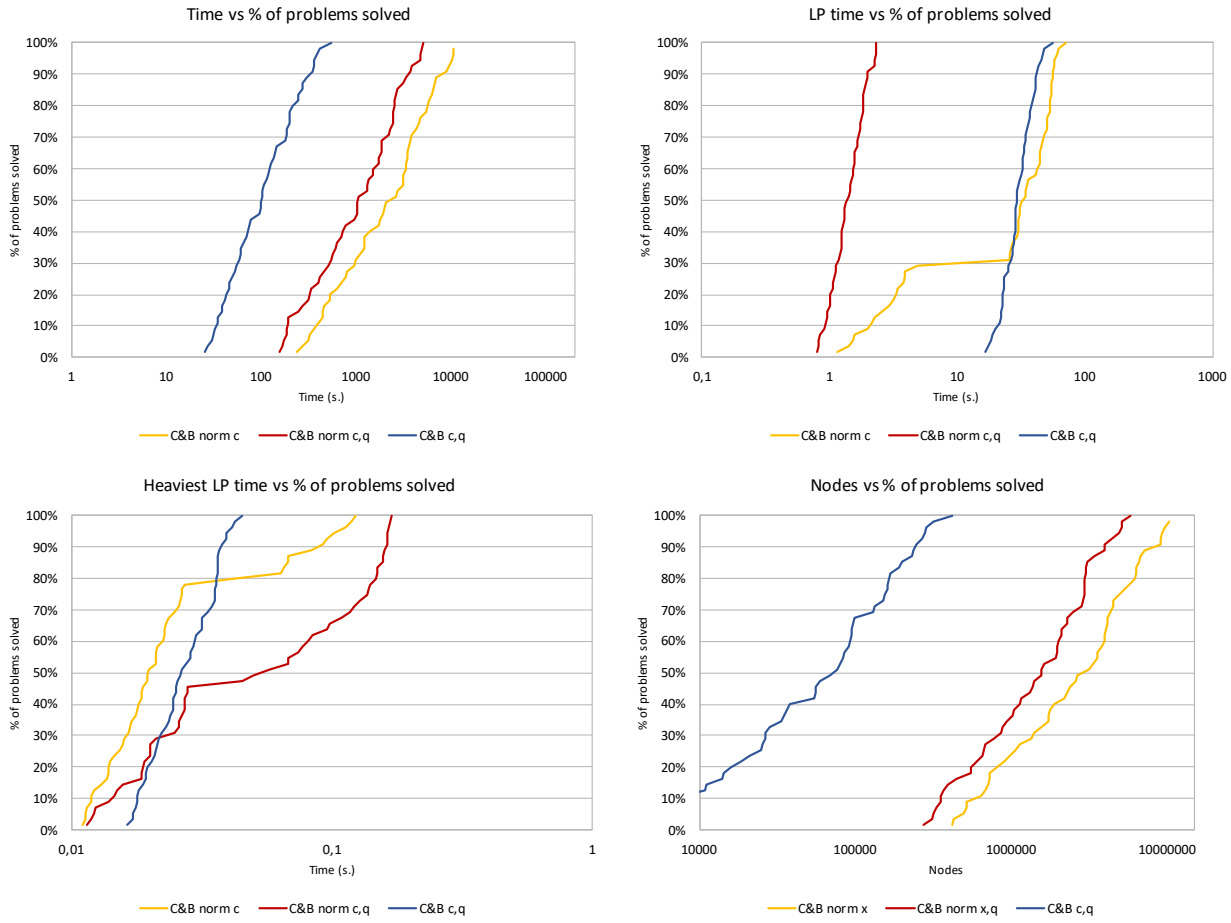

Figure 7.36: Performance profile graphs over security instances A.

We can see that (C&B$_{c,q}$) is the fastest algorithm, and (C&B$_c$norm) is the slowest. With regard to relaxation time, we observe that (C&B$_{c,q}$norm) has the fastest LP time, and (C&B$_c$norm) has the slowest in the most of the instances. If we compare the solution time of the last linear problem of the root node, we see that (C&B$_c$norm) produces relaxations easier to solve than others in the most of instances, and (C&B$_{c,q}$norm) does the opposite. Finally, we can see that (C&B$_{c,q}$) explores the smallest number of nodes. With this information, we can say that the linear relaxations of (C&B$_{c,q}$) are better than for other algorithms, and this causes less nodes explored that make up for the slower relaxations, which leads to better solution times. The linear relaxations of the different algorithms are drawn in figure (7.37), where (C&B$_{c,q}$) presents relaxations much better than others.

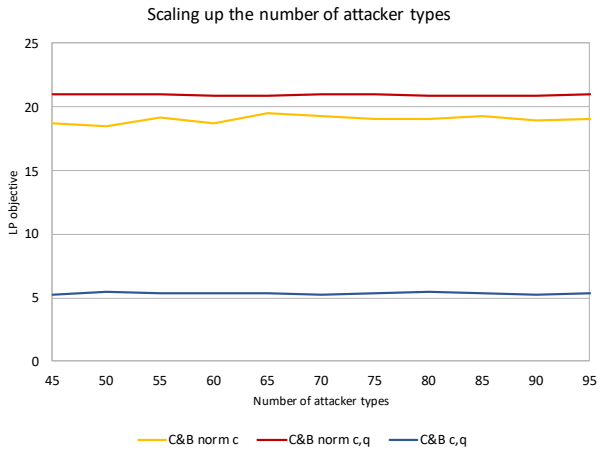

Figure 7.37: Average linear relaxation objective scaling up the number of attacker types.

Instances B

We compare our best configured algorithms against (C&B$_{c,q}$) algorithm, which was the best in [19] for this set of instances. Its best configuration is: 0.7 stabilization, advanced LB heuristic and UB heuristic with parameter $0.01 - 0.25$. This algorithm is the same used for security instances A.

Figure (7.38) shows the average solution time for the different algorithms, scaling up the number of targets. We can see that (C&B$_{c,q}$) taken from [19] is the fastest solution method for every number of targets.

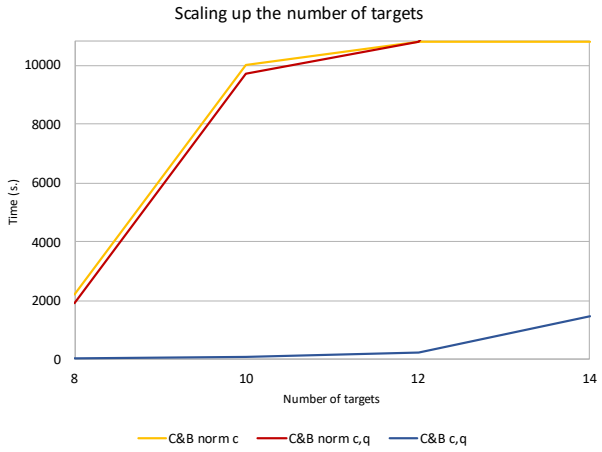

Figure 7.38: Average solution time scaling up the number of targets.

Figure (7.39) presents performance profile graphs to compare solution time, LP solution time, solution time of the last linear problem of the root node relaxation and nodes explored.

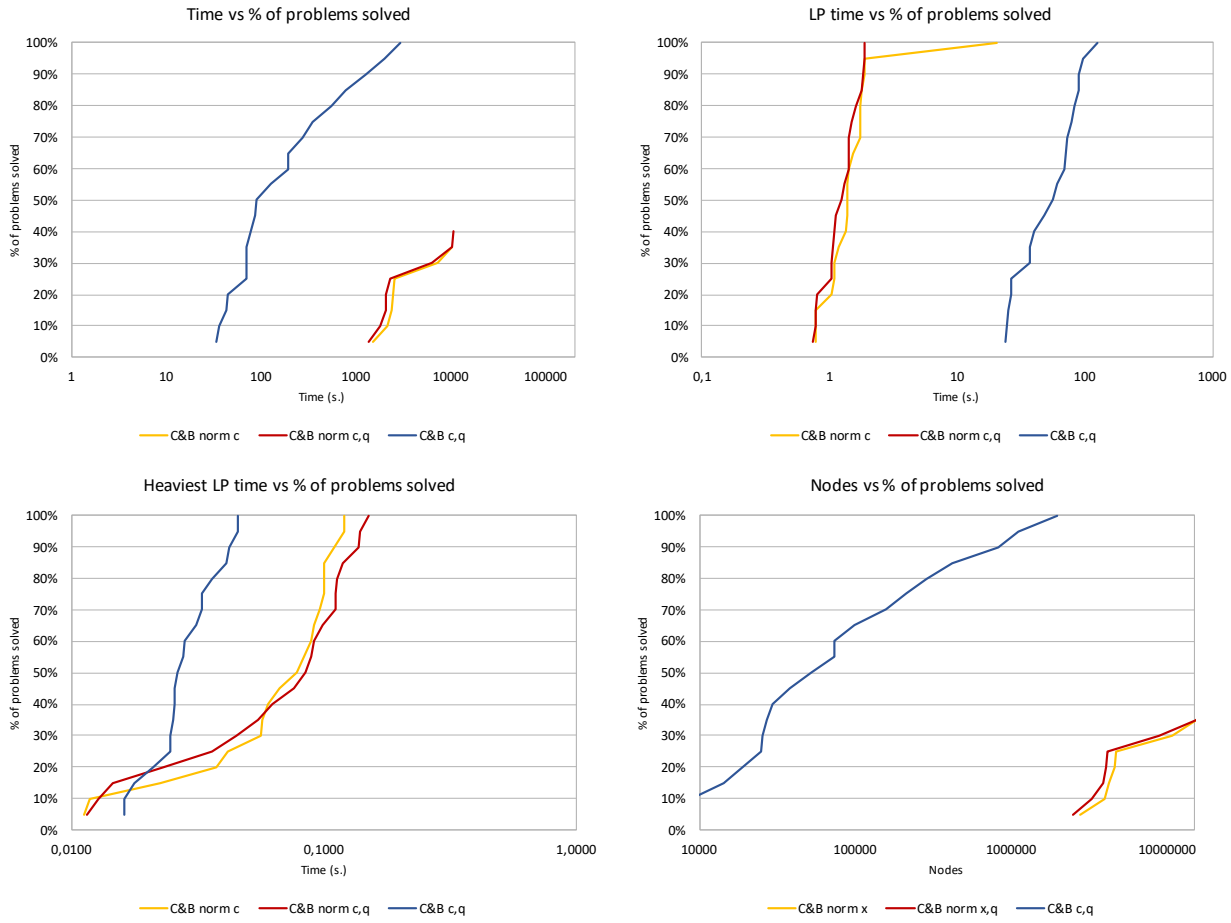

Figure 7.39: Performance profile graphs over security instances B.

We can see that (C&B$_{c,q}$) is the fastest algorithm, and (C&B$_c$norm) is the slowest, even both normalized algorithms just solved 40% of the problems. With regard to relaxation time, we observe that (C&B$_{c,q}$norm) has the fastest LP time followed very close by (C&B$_c$norm). If we compare the solution time of the last linear problem of the root node, we see that (C&B$_{c,q}$) produces relaxations easier to solve than others in the most of instances. Finally, we can see that (C&B$_{c,q}$) explores the smallest number of nodes. With this information, we can say that the linear relaxations of (C&B$_{c,q}$) are better than for other algorithms, and this causes less nodes explored that make up for the slower relaxations, which leads to better solution times. The linear relaxations are drawn in figure (7.40), where is clear that (C&B$_{c,q}$) has much better relaxations than our algorithms.

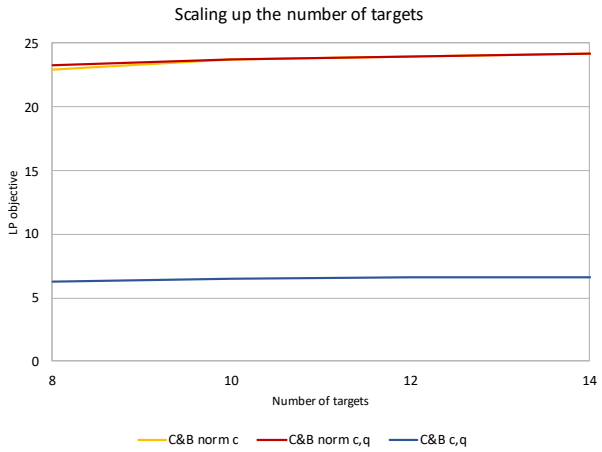

Figure 7.40: Average linear relaxation objective scaling up the number of targets.

Chapter 8

Conclusion

In this thesis we studied branching algorithms based in Benders decomposition, for solving general and security Stackelberg games. Our algorithms are based in the idea of [19], who mixed different general and security Stackelberg games formulations for creating efficient cut & branch algorithms by using Benders decomposition for generating the cuts, keeping fractional variables in master problem. These algorithms outperformed others existing until then.

We first studied a branch & cut algorithm, for general and for security setting, generating optimality and feasibility cuts based in Benders decomposition, in the whole search tree. We used as master problem the linear relaxation of a fast formulation, ($D2_{x,q,s,f}$) or ($ERASER_{c,q,s,f}$), and the dual of subproblems of (MIP-p-$G_{x,q,z}$) and (MIP-p-$S_{c,q,z}$) for generating the cuts. In our setting, we kept integer variables in master problem and generated cuts across the wholw search tree. This algorithm solved less than half of the general instances and 0% of security instances in the time limit. Therefore, its performance was very poor compared against the cut & branch algorithms created by [19], which kept different combinations of fractional variables in master problem and generated cuts only in the root node, and solved all the instances in time limit. We considered two types of instances for each Stackelberg setting: scaling up the number of follower types or the number of pure strategies, for general; and scaling up the number of attacker types or the number of targets, for security.

Furthermore, we studied a normalized branch & cut algorithm. This algorithm is a variant of the standard branch & cut studied first. The normalization consisted of converting the standard Benders cut generation conditions, optimality and feasibility, in one polyhedron that included both. Then, we normalized it by bounding the objective function and adding some constraints to dual variables. This algorithm outperforms the previous branch & cut, but its performance is still poor against [19].

After normalization of the branch & cut algorithms and the subtle performance improvement, we decided to normalize the cut & branch algorithms of [19]. Two similar algorithms with different fractional variables in master problem. These algorithms had so much better performance than the previous branch & cut algorithms. Therefore, we choose them for

trying different heuristics and compare against [19]. We studied the application of different heuristics in the chosen algorithms, taken from [19]. Cut loop stabilization heuristics improved the performance of our algorithms in some types of instances; lower bound enhancing heuristics didn't help to improve any algorithm performance in any type of instance; and upper bound enhancing heuristics didn't help either.

After configuring our algorithms with best heuristics, we carried out the comparison across different set of instances, and the algorithms of [19] were much better:

- Our algorithms were much slower to solve.
- Our algorithms had faster linear relaxations, but the competitor relaxations were tighter.
- Our algorithms visited larger number of nodes than competitor.

8.1 Other results and research proposals

The normalization helped to improve the branch & cut algorithms, but not the cut & branch. Recall that normalization reduces the total number of cuts generated and, as branch & cut generate cuts in the whole branching tree, it has a bigger impact in this type of algorithm.

We also found that our normalized algorithms produced very weak linear relaxations against the standard cut & branch algorithms compared in this thesis. In particular, upper bounds from these relaxations were much bigger than others. It would be interesting to study the way to strengthen the relaxations of the normalized general and security Stackelberg versions.

We used the heuristics proposed by [19] that worked in his algorithms. It would be interesting to research other heuristics that could help better in normalized algorithms.

Bibliography

[1] Bo An, Fernando Ordóñez, Milind Tambe, Eric Shieh, Rong Yang, Craig Baldwin, Joseph DiRenzo III, Kathryn Moretti, Ben Maule, and Garett Meyer. A deployed quantal response-based patrol planning system for the u.s. coast guard. *Interfaces*, 43(5):400–420, 2013.

[2] Robert Aumann. Acceptable points in general cooperative n-person games. In Albert Tucker and Robert Luce, editors, *Contributions to the theory of games, volume IV*, Annals in mathematics Studies, 40, pages 287–324. Princeton University Press, Princeton, New Jersey, 1959.

[3] Robert Aumann, Michael Maschler, and Richard Stearns. *Repeated games with incomplete information.* MIT Press, Cambridge, Massachusetts, 1995.

[4] Egon Balas, Sebastián Ceria, and Gérard Cornuéjols. A lift and project cutting plane algorithm for mixed 0-1 programs. *Mathematical Programming*, 58:295–324, 1993.

[5] Egon Balas, Sebastián Ceria, Gérard Cornuéjols, and N. Natraj. Gomory cuts revisited. *Operations Research Letters*, 19:1–9, 1996.

[6] Jonathan F. Bard. Convex two-level optimization. *Mathematical Programming*, 40:15–27, 1998.

[7] Nils Baumgärtner, Matthias Leisin, Björm Bahl, Maike Hennen, and André Bardow. Rigorous synthesis of energy systems by relaxation and time-series aggregation to typical periods. In Mario Eden, Marianthi Ierapetritou, and Gavin Towler, editors, *Proceedings of the 13th International Symposium on Process Systems Engineering - PSE2018*, pages 793–798, San Diego, California, 2018. Elsevier.

[8] T. Başar and G. J. Olsder. *Dynamic noncooperative game theory.* Academic Press, San Diego, CA, 1995.

[9] Walid Ben-Ameur and José Neto. Acceleration of cutting-plane and column generation algorithms: applications to network design. *Networks*, 49(1):3–17, 2007.

[10] Omar Ben-Ayed and Charles E. Blair. Computational difficulties of bilevel linear programming. *Operations Research*, 38(3):556–560, 1990.

[11] J. F. Benders. Partitioning procedures for solving mixed-variables programming prob-

lems. *Numerische Mathematik*, 4:238–252, 1962.

[12] Joseph Bertrand. Theorie mathématique de la richesse sociale. *Journal des Savants*, 1883:499–508, 1883.

[13] Dimitris Bertsimas and John N. Tsitsiklis. *Introduction to linear optimization*. Athena Scientific, Belmont, Massachusetts, 1997.

[14] Dimitris Bertsimas and Robert Weistmantel. *Optimization over integers*. Dynamic Ideas, Belmont, Massachusetts, 2005.

[15] Michael Bloem, Tansu Alpcan, and Tamer Başar. A stackelberg game for power control and channel allocation in cognitive radio networks. In *Proceedings of the 2nd International Conference on Performance Evaluation Methodologies and Tools - ValueTools 2007*, pages 4:1–4:9, Brussels, Belgium, 2007. Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering.

[16] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, New York, New York, 2004.

[17] Jerome Bracken and James T. McGill. Mathematical programs with optimization problems in the constraints. *Operations Research*, 21:37–44, 1973.

[18] Jean Cardinal, Martine Labbé, and Stefan Langerman. Pricing geometric transportation networks. *International Journal of Computational Geometry & Applications*, 19(6):507–520, 2009.

[19] Carlos Casorrán. *Formulations and algorithms for general and security Stackelberg games*. PhD thesis, Université Libre de Bruxelles and Universidad de Chile, 2018.

[20] Carlos Casorrán, Bernard Fortz, Martine Labbé, and Fernando Ordóñez. A study of general and security stackelberg game formulations. *European Journal of Operational Research*, 278(3):855–868, 2019.

[21] Yunfei Chu and Fengqi You. Integrated scheduling and dynamic optimization by stackelberg game: bilevel model formulation and efficient solution algorithm. *Industrial & Engineering Chemistry Research*, 53:5564–5581, 2014.

[22] Vincent Conitzer and Dmytro Korzhyk. Commitment to correlated strategies. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence - AAAI 2011*, pages 632–637, San Francisco, California, 2011. AAAI Press.

[23] Cécile Cordier, Hughes Marchand, Richard Laundy, and Laurence Wolsey. Bc-opt: a branch-and-cut code for mixed integer programs. *Mathematical Programming*, 86(2):335–353, 1999.

[24] Antoine-Augustin Cournot. *Recherches sur les principes mathématiques de la théorie des richesses*. Hachette Livre, París, France, 1838.

[25] George Dantzig and B. Eaves. Fourier-motzkin elimination and its dual. *Journal of Combinatorial Theory (A)*, 14:288–297, 1973.

[26] Stephan Dempe and Joydeep Dutta. Is bilevel programming a special case of a mathematical program with complementarity constraints? *Mathematical Programming*, 131:37–48, 2012.

[27] Julus Farkas. Theorie der einfachen ungleichungen. *Journal fr die reine und angewandte Mathematik*, 124:1–27, 1902.

[28] Mateo Fischetti, Ivana Ljubić, and Markus Sinnl. Redesigning benders decomposition for large-scale facility location. *Management Science*, pages 1–17, 2016.

[29] Matteo Fischetti, Domenico Salvagnin, and Arrigo Zanette. Minimal infeasible subsystems and benders cuts, 2008.

[30] Matteo Fischetti, Domenico Salvagnin, and Arrigo Zanette. A note on the selection of benders' cuts. *Math. Program.*, 124:175–182, 2010.

[31] John Gleeson and Jennifer Ryan. Identifying minimally infeasible subsystems of inequalities. *ORSA Journal on Computing*, 2(1):61–63, 1990.

[32] Ralph Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 64:275–278, 1958.

[33] Karla Hoffman and Ted Ralphs. Integer and combinatorial optimization. *COR@L Technical Report 12T-020*, 2012.

[34] M. Hoseininia, M. Seyyed, F. Didehvar, and A. Haghi. Inventory competition in a multi channel distribution system: the nash and stackelberg game. *Scientia Iranica E*, 20(3):846–854, 2013.

[35] Robert G. Jeroslow. The polynomial hierarchy and a simple model for competitive analysis. *Mathematical Programming*, 32(2):146–164, 1985.

[36] Christopher Kiekintveld, Manish Jain, Jason Tsai, James Pita, Fernando Ordóñez, and Milind Tambe. Computing optimal randomized resource allocations for massive security games. In *Proceedings of the 8th International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1, AAMAS 2009*, pages 689–696, Budapest, Hungary, 2009. International Foundation for Autonomous Agents and Multiagent Systems.

[37] Harold Kuhn. Extensive games. *Proceedings of the National Academy of Science of the United States of America*, 36:570–576, 1950.

[38] Martine Labbé, Patrice Marcotte, and Gilles Savard. A bilevel model of taxation and its application to optimal highway pricing. *Management Science*, 44(12-part-1):1608–1622, 1998.

[39] Alisa Land and Alison Doig. An automatic method of solving discrete programming

problems. *Econometrica*, 28(3):497–520, 1960.

[40] G. Leitman. On generalized stackelberg strategies. *Journal of Optimization Theory and Applications*, 26(4):637–643, 1978.

[41] Richard Lewontin. Evolution and theory of games. *Journal of Theoretical Biology*, 1:382–703, 1961.

[42] Robert McNaughton. Scheduling with deadlines and loss functions. *Maganement Science*, 6(1):1–12, 1959.

[43] John Nash. *Non.cooperative games*. PhD thesis, Departament of Mathematics, Princeton University, 1950.

[44] George Nemhauser and Laurence Wolsey. *Integer and combinatorial optimization*. John Wiley & sons, New York, 1988.

[45] Martin J. Osborne and Ariel Rubinstein. *A course in game theory*. The MIT Press, Oxford, Massachusetts and London, England, 1994.

[46] Praveen Paruchuri, Jonathan P. Pearce, Janusz Marecki, Milind Tambe, Fernando Ordóñez, and Sarit Kraus. Playing games for security: an efficient exact algorithm for solving bayesian stackelberg games. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 2, AAMAS 2008*, pages 895–902, Richland, SC, 2008. International Foundation for Autonomous Agents and Multiagent Systems.

[47] Praveen Paruchuri, Jonathan P. Pearce, Milind Tambe, Fernando Ordóñez, and Sarit Kraus. An efficient heuristic approach for security against multiple adversaries. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS 2007*, pages 1–8, Honolulu, Hawaii, 2007. International Foundation for Autonomous Agents and Multiagent Systems.

[48] Heejun Roh, Hoorin Park, Cheoulhoon Jung, and Wonjun Lee. A stackelberg game for spectrum investment and pricing in cooperative cognitive radio networks. In *Proceedings of the ACM CoNEXT Student Workshop 2011*, pages 8:1–2, New York, New York, 2011. ACM.

[49] Ulrich Schwalbe and Paul Walker. Zermelo and the early history of game theory. *Games and Economic Behavior*, 34:123–137, 2001.

[50] Lloyd Shapley and Martin Shubik. A method for evaluating the distribution of power in a committee system. *American Political Science Review*, 48:787–792, 1954.

[51] Hanif D. Sherali and Warren P. Adams. A hierarchy of relaxations and convex hull characterizations for mixed-integer zero-one programming problems. *Discrete Applied Mathematics*, 52:83–106, 1994.

[52] Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. A review on bilevel optimization: from

classical to evolutionary approaches and applications. *IEEE Transactions on evolutionary computation*, 22(2):276–295, 2018.

[53] Bernard W. Taylor. *Introduction to management science*. Prentice Hall, Upper Saddle River, New Jersey, 2006.

[54] Ángel Tenorio and Ana Martín. A trip down the story of game theory. *Boletín de Matemáticas*, 22(1):77–95, 2015.

[55] John von Neumann. Zur theorie der gesellschaftsspiele. *Mathematische Annalen*, 100:295–320, 1928.

[56] John von Neumann and Oskar Morgenstern. *Theory of games and economic behavior*. Princeton University Press, Princeton, 1944.

[57] Heinrich von Stackelberg. *Marktform und gleichgewicht*. Springer-Verlag, Berlin, 1934.

[58] Heinrich von Stackelberg. *The theory of the market economy*. Oxford University Press, Oxford, 1952.

[59] Bernhard von Stengel and Shmuel Zamir. Leadership with commitment to mixed strategies. *CDAM Research Report LSE-CDAM-2004-01*, 2004.

[60] Xiao-Cheng Wang, Xin-Ping Guan, Qiao-Ni Han, Zhi-Xin Liu, and Kai Ma. A stackelberg game for spectrum leasing in cooperative cognitive radio networks. *International Journal of Automation and Computing*, 10(2):125–133, 2013.

[61] Laurence Wolsey. *Integer programming*. John Wiley & sons, New York, 1998.

[62] Zhengyu Yin, Dmytro Korzhyk, Christopher Kienkintveld, Vincent Conitzer, and Milind Tambe. Stackelberg vs. nash in security games: an extended investigation on interchangeability, equivalence, and uniqueness. *Journal of Artificial Intelligence Research*, 41:297–327, 2011.

[63] Yugang Yu, George Q. Huang, and Liang Liang. Stackelberg game-theoretic model for optimizing advertising, pricing and inventory policies in vendor inventory (vmi) production supply chains. *Computer & Industrial Engineering*, 57:368–382, 2009.

[64] Ernst Zermelo. Über eine anwendung der mengenlehre auf die theorie des schachspiels. In *Proceedings of the 5th International Congress Mathematicians*, pages 501–504, Cambridge, 1913. Cambridge University Press - Volume 2.

Appendix A

Switching roles between objective function and normalization condition

We demonstrate the equivalence of two problems in normalized Benders decomposition approach: alternative polyhedron and separation problem. We first show a demonstration for a particular approach: general case with q in master problem. After doing that, we show a generalization that can be applied to remaining normalized approaches.

General case with q in master problem

Let's consider the alternative polyhedron:

$$
\begin{aligned}
\min_{\alpha,\beta,\gamma,\lambda} \quad & \sum_{j\in J}\sum_{k\in K}\beta_j^k + \lambda \\
\text{s.t} \quad & \sum_{k\in K}\alpha_{\mathrm{i}}^k = 0 \quad \forall \mathrm{i} \in I, \\
& \alpha_{\mathrm{i}}^k + \beta_j^k + \sum_{l\in J: l\neq j}(C_{\mathrm{ij}}^k - C_{\mathrm{il}}^k)\gamma_{jl}^k + \\
& \qquad \lambda\pi^k R_{\mathrm{ij}}^k \geq 0 \quad \forall \mathrm{i} \in I, \forall j \in J, \forall k \in K, \\
& \gamma_{jl}^k \leq 0 \quad \forall \mathrm{i} \in I, \forall j \in J, \forall k \in K, \\
& \lambda \leq 0 \\
& \sum_{j\in J}\sum_{k\in K}\beta_j^k q_j^{k^*} + \lambda\theta^* = -1.
\end{aligned}
\tag{A.1}
$$

and the separation problem:

$$
\begin{aligned}
\min_{\alpha,\beta,\gamma,\lambda} \quad & \sum_{j\in J}\sum_{k\in K}\beta_j^k q_j^{k^*} + \lambda\theta^* \\
\text{s.t} \quad & \sum_{k\in K}\alpha_{\mathrm{i}}^k = 0 \quad \forall \mathrm{i} \in I, \\
& \alpha_{\mathrm{i}}^k + \beta_j^k + \sum_{l\in J: l\neq j}(C_{\mathrm{i}j}^k - C_{\mathrm{i}l}^k)\gamma_{jl}^k + \\
& \qquad \lambda\pi^k R_{\mathrm{i}j}^k \geq 0, \quad \forall \mathrm{i} \in I, \forall j \in J, \forall k \in K, \\
& \gamma_{jl}^k \leq 0 \quad \forall \mathrm{i} \in I, \forall j \in J, \forall k \in K, \\
& \lambda \leq 0, \\
& \sum_{j\in J}\sum_{k\in K}\beta_j^k + \lambda = -1.
\end{aligned}
\tag{A.2}
$$

We demonstrate the equivalence between both problems of section §5.2.2, where the normalization condition is switched with the objective function.

We know that two optimization problems are equivalent if a solution for one of them can be used to construct a solution for the other.

Let's consider a solution for problem (A.1):

$$(\beta^*, \lambda^*) = (\tilde{\beta}, \tilde{\lambda})$$

This yields an objective value of

$$\sum_{j\in J}\sum_{k\in K}\tilde{\beta}_j^k + \tilde{\lambda}, \tag{A.3}$$

and the normalization condition gets

$$\sum_{j\in J}\sum_{k\in K}\tilde{\beta}_j^k q_j^{k^*} + \tilde{\lambda}\theta^* = -1. \tag{A.4}$$

Now consider a solution for problem (A.2), as a function of the first problem solution:

$$(\beta^*, \lambda^*) = (\hat{\beta}, \hat{\lambda}) = -\left(\frac{1}{\sum_{j\in J}\sum_{k\in K}\tilde{\beta}_j^k + \tilde{\lambda}}\right)(\tilde{\beta}, \tilde{\lambda})$$

This yields the following objective value:

$$-\sum_{j\in J}\sum_{k\in K}\left(\frac{1}{\sum_{j\in J}\sum_{k\in K}\tilde{\beta}_j^k + \tilde{\lambda}}\right)\tilde{\beta}_j^k q_j^{k^*} - \left(\frac{1}{\sum_{j\in J}\sum_{k\in K}\tilde{\beta}_j^k + \tilde{\lambda}}\right)\tilde{\lambda}\theta^*$$

and with some algebra we get:

$$-\frac{\sum_{j\in J}\sum_{k\in K}\tilde{\beta}_j^k q_j^{k^*} + \tilde{\lambda}\theta^*}{\sum_{j\in J}\sum_{k\in K}\tilde{\beta}_j^k + \tilde{\lambda}}.$$

The upper side gets equal to -1 for the normalization condition (A.4). This gives the final objective value:

$$\frac{1}{\sum_{j\in J}\sum_{k\in K}\tilde{\beta}_j^k + \tilde{\lambda}}. \tag{A.5}$$

We now try the feasibility of the last constraint in the separation problem (A.2):

$$-\sum_{j\in J}\sum_{k\in K}\left(\frac{1}{\sum_{j\in J}\sum_{k\in K}\tilde{\beta}_j^k + \tilde{\lambda}}\right)\tilde{\beta}_j^k - \left(\frac{1}{\sum_{j\in J}\sum_{k\in K}\tilde{\beta}_j^k + \tilde{\lambda}}\right)\tilde{\lambda},$$

and with some algebra we get

$$-\left(\frac{\sum_{j\in J}\sum_{k\in K}\tilde{\beta}_j^k + \tilde{\lambda}}{\sum_{j\in J}\sum_{k\in K}\tilde{\beta}_j^k + \tilde{\lambda}}\right) = -1, \tag{A.6}$$

which makes the constraint feasible.

Generalization

Let's consider general Benders decomposition approach from the generic problem used in [30]:

$$\begin{aligned} \min_{x,\,y} \quad & c^T x + \mathrm{d}^t y \\ \text{s.t} \quad & Ax \geq b, \\ & Tx + Qy \geq r, \\ & x \in \mathbb{Z}_+^n, y \in \mathbb{R}_+^t. \end{aligned} \tag{A.7}$$

We have the following alternative polyhedron:

$$\begin{aligned} \max_{\pi,\,\pi_0} \quad & \sum_{\mathrm{i}=1}^{m} w_{\mathrm{i}}\pi_{\mathrm{i}} + w_0\pi_0 \\ \text{s.t} \quad & \pi^T Q \leq \pi_0 \mathrm{d}^T, \\ & \pi^T(r - Tx^*) - \pi_0\eta^* = 1, \\ & (\pi, \pi_0) \geq 0. \end{aligned} \tag{A.8}$$

We show that problem (A.8) is equivalent to separation problem (A.9):

$$\begin{aligned} \max_{\pi,\,\pi_0} \quad & \pi^T(r - Tx^*) - \pi_0\eta^* \\ \text{s.t} \quad & \pi^T Q \leq \pi_0 \mathrm{d}^T, \\ & \sum_{\mathrm{i}=1}^{m} w_{\mathrm{i}}\pi_{\mathrm{i}} + w_0\pi_0 = 1, \\ & (\pi, \pi_0) \geq 0. \end{aligned} \tag{A.9}$$

Let's consider a solution for problem (A.8):

$$(\pi^*, \pi_0^*) = (\tilde{\pi}, \tilde{\pi_0})$$

This yields an objective value of

$$\sum_{\mathrm{i}=1}^{m} w_{\mathrm{i}}\tilde{\pi}_{\mathrm{i}} + w_0\tilde{\pi}_0 \tag{A.10}$$

and the normalization condition gets:

$$\tilde{\pi}^T(r - Tx^*) - \tilde{\pi}_0\eta^* = 1 \tag{A.11}$$

Now consider a solution for problem (A.9), as a function of the first problem solution:

$$(\pi^*, \pi_0^*) = (\hat{\pi}, \hat{\pi}_0) = \left(\frac{1}{\sum_{\mathrm{i}=1}^{m} w_{\mathrm{i}}\tilde{\pi}_{\mathrm{i}} + w_0\tilde{\pi}_0}\right)(\tilde{\pi}, \tilde{\pi}_0)$$

This yields the following objective value:

$$\frac{\tilde{\pi}^T(r - Tx^*) - \tilde{\pi}_0\eta^*}{\sum_{\mathrm{i}=1}^{m} w_{\mathrm{i}}\tilde{\pi}_{\mathrm{i}} + w_0\tilde{\pi}_0}.$$

The upper side gets equal to 1 for the normalization condition (A.11). This gives the final objective value:

$$\frac{1}{\sum_{\mathrm{i}=1}^{m} w_{\mathrm{i}}\tilde{\pi}_{\mathrm{i}} + w_0\tilde{\pi}_0}. \tag{A.12}$$

We now try the feasibility of the last constraint in separation problem (A.9)

$$\sum_{\mathrm{i}=1}^{m} \left(\frac{1}{\sum_{\mathrm{i}=1}^{m} w_{\mathrm{i}}\tilde{\pi}_{\mathrm{i}} + w_0\tilde{\pi}_0} \right) \tilde{\pi}_{\mathrm{i}} + \left(\frac{1}{\sum_{\mathrm{i}=1}^{m} w_{\mathrm{i}}\tilde{\pi}_{\mathrm{i}} + w_0\tilde{\pi}_0} \right) \tilde{\pi}_0,$$

and with some algebra we get:

$$\frac{\sum_{\mathrm{i}=1}^{m} w_{\mathrm{i}}\tilde{\pi}_{\mathrm{i}} + w_0\tilde{\pi}_0}{\sum_{\mathrm{i}=1}^{m} w_{\mathrm{i}}\tilde{\pi}_{\mathrm{i}} + w_0\tilde{\pi}_0} = 1, \tag{A.13}$$

which makes the constraint feasible.