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

EFFICIENT ALGORITHMS FOR RISK AVERSE OPTIMIZATION

TESIS PARA OPTAR AL GRADO DE DOCTOR EN SISTEMAS DE INGENIERÍA

RENAUD PIERRE CHICOISNE

PROFESOR GUÍA:
FERNANDO ORDÓÑEZ PIZARRO

PROFESOR CO-GUÍA:
DANIEL ESPINOZA GONZÁLEZ

MIEMBROS DE LA COMISIÓN:
EDUARDO MORENO ARAYA
MAGED DESSOUKY

SANTIAGO DE CHILE
JULIO 2015

Resumen

Muchos problemas de decisión industriales o logísticos pueden ser vistos como problemas de optimización y para muchos de ellos no es razonable ocupar datos deterministas. Como veremos en este trabajo en el contexto de despachos de emergencia o de planificación de seguridad, las condiciones reales son desconocidas y tomar decisiones sin considerar esta incertidumbre pueden llevar a resultados catastróficos. La teoría y la aplicación de optimización bajo incertidumbre es un tema que ha generado un amplio área de investigación. Sin embargo, aún existen grandes diferencias en complejidad entre optimización determinista y su versión incierta. En esta tesis, se estudian varios problemas de optimización con aversión al riesgo con un énfasis particular en el problema de camino más corto (RASP), problemas estocásticos en redes en general y juegos de seguridad de Stackelberg.

Para obtener distribuciones de tiempos de viaje precisos sobre una red vial a partir de datos GPS del sistema de tránsito, se presenta una revisión de los métodos existentes para proyectar trayectorias GPS y se definen dos nuevos algoritmos: Uno que permite la proyección de datos óptima con respecto a una medida de error convenientemente definida (MOE), y un método heurístico rápido que permite proyectar grandes cantidades de datos de manera contínua (MMH). Se presentan resultados computacionales en redes reales y generadas de gran tamaño. Luego, se desarrollan algoritmos eficientes para problemas de ruteo con aversión al riesgo utilizando métodos de Sample Average Approximation, técnicas de linealización y métodos de descomposición. Se estudian la medida de riesgo entrópica y el Conditional Value at Risk considerando correlaciones entre las variables aleatorias. Se presentan resultados computacionales prometedores en instancias generadas de tamaño mediano. Sin embargo, la naturaleza combinatorial de los problemas los vuelve rápidamente intratable a medida que el tamaño del problema crece. Para hacer frente a esta dificultad computacional, se presentan nuevas formulaciones para problemas en redes difíciles, que tienen un menor número de variables enteras. Estas formulaciones ayudan a derivar esquemas de brancheo que se aprovechan de la estructura especial de las formulaciones propuestas. Se muestra cómo aplicar estas ideas a los conjuntos de camino simple y de circuito hamiltoniano en redes generales, así como los conjuntos de camino simple y de corte en grafos dirigidos acíclicos (DAG). Este trabajo preliminar muestra ideas prometedoras para resolver problemas difíciles. Finalmente, se exploran las implicaciones de los métodos algorítmicos y las formulaciones desarrolladas para resolver RASP en un área diferente. Se presentan nuevas formulaciones y enfoques de resolución para juegos de seguridad de Stackelberg cuando el defensor es averso al riesgo con respecto a la estrategia del atacante. Esto se puede resolver de manera polinomial cuando se enfrenta a un adversario y resolviendo un problema de optimización convexa en números enteros cuando el defensor enfrenta varios tipos de adversarios.

Abstract

Many decision problems in industry or logistics can be viewed as optimization problems and it is commonly accepted that for a number of them it is unreasonable to assume deterministic data. In emergency dispatching or security planning, the unknown actual conditions are a significant source of uncertainty and wrong decisions can lead to dire situations. The theory and application of optimization under uncertainty has been a source of substantial research. Nevertheless, there still exist huge gaps in difficulty between deterministic optimization and its version with uncertainty. In this thesis, we study several risk averse optimization problems, with a particular emphasis on risk averse shortest path (RASP), stochastic problems in networks in general, and risk averse Stackelberg security games.

To obtain accurate travel time distribution data on road networks from real world GPS transit system trajectory data, we first present an overview of existing methods of projecting GPS trajectories and define two new algorithms: One allowing optimal data projection with respect to a suitably defined error measure, and a fast heuristic method that enables online data projection for large datasets. We present computational results on large scale networks (real and generated). We then develop efficient algorithms for RASP problems using sample average approximation, linearization techniques and decomposition methods. We studied the Entropic risk measure and the Conditional Value at Risk and considered correlations between the random variables. We present promising computational results on medium sized generated instances. Nevertheless, as the problem size grows we show that the problem can become intractable. To address this computational difficulty, we present new formulations for difficult network problems that have fewer integer variables. These formulations help derive constraint branching schemes that take advantage of the special structure of the new formulations. We show how to apply these formulations and associated branching schemes to the st-path and the Hamiltonian circuit sets in general networks as well as the st-path and st-cut sets in Directed Acyclic Graphs (DAGs). This preliminary work shows promising ideas for the development of practical resolution schemes for hard network problems. Finally, we explore the implications of the algorithmic methods and formulations developed to solve RASP in a whole different domain. We present novel formulations and solution approach for Stackelberg security games when the defender is risk averse with respect to the attacker strategy. This leads to a polynomial method when facing one adversary and a convex Mixed Integer Non Linear Programming (MINLP) formulation when the defender faces several types of adversaries.

To my father and grandmother.

Acknowledgements

For the past five years, I have been a PhD student at the DSI doctoral program of the University of Chile in Santiago. I have carried my research in the Operations Research group, which turned out to be a fertile basis for my investigation on the combination of mathematical programming and real life optimization. I want to thank Fernando Ordóñez and Daniel Espinoza whose guidance was more than useful to go on with the research and helped me personally more than once in the day-to-day life. At the Department of Industrial Engineering, I have benefited greatly from the AGCO seminars and the knowledge of its top-notch members. In particular, I thank all the professors of the University of Chile and the University Adolfo Ibáñez who gave me five minutes countless times for all the weird questions I had in mind. A special thank to Roberto Cominetti who was always available for a short chat.

As well, all my gratitude goes to all the people with whom I shared an office, a course or a common project, in particular to Alvaro Echeverría, Francisco Muñoz and the three Victors, I would like to thank the long list of roommates I had during those five PhD years: Tchomas, Nico, Ale, Pelao Galvez, Roberto, Cacho, Dani, Steffi, Alex, Mathieu, Panchopin and Jimador among others. They were a great support all the times I got flooded with the thesis workload. It was a great pleasure to be the teacher of a lot of skillful students like Ignacio, Matías, Ángela or Nicolás, who are beginning what I am finishing right now.

I thank my family and friends in France that accompanied me all the time in spite of the thousands of kilometers that separated us, and Javiera and the two felines who were always here when coming back home. Last but not least, a special thought to my father and my hundred springs old grandmother who both left us during these five years.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

Many real life problems require making decisions in uncertain or changing environments. To address this there exists a large literature of research on optimization models and methods that consider parameter uncertainty. We can name, among others, finding the cheapest routing of a fleet of trucks delivering packages from depots to customers under uncertain travel times ([50]), locating facilities in the most efficient way given variable customers demand ([40]) or determining stock levels depending on a variable demand over time ([32, 41]). All these examples require a thorough understanding of the structure of the travel times or the demand pattern which are inherently uncertain in the general case. The same goes in a game theoretical context ([60]) where a player - although knowing the mixed strategy of the other agents - is unaware of the exact decisions of the adversaries in the general case. Consequently, we cannot assume *a priori* that we know the exact duration of travel through some road section or that we have perfect knowledge of a future demand or the adversaries actions in a game. Consequently, in many cases problem parameters must be represented by a random variable with some probability distribution probability distribution rather than a single number. We can note that any problem assuming distributions is at least as difficult as a problem assuming deterministic data as the latter is a special case of the former.

This Ph.D. thesis began with a work done for the redesign of the dispatching system for the Santiago Fire Department (SFD) that is in use since December 2012 ([34]). The previous system used static dispatching rules, matching every possible emergency location with predetermined fire stations. The static nature of this assignment and its independence on the hour of the emergency and the network topology could lead to inefficient dispatching rules. As it is crucial to have the fastest possible response, we built a network representation of the Santiago transportation system and decided which fire stations must dispatch firemen to an emergency according to the shortest paths in this network. We needed new algorithms to efficiently project GPS data from the transit system to estimate congestion on roads as this new system required accurate travel time estimations on all road segments in Santiago at different times of day. Furthermore, as delays in emergency services can lead to dire consequences, it is reasonable to select shortest paths to an emergency that are risk averse.

Moreover, deterministic Shortest Paths Problems (SPP) are polynomially solvable in theory and fast to solve in practice ([3]) so we can easily observe the gap in difficulty when including uncertainty. Indeed, when including risk aversion into this polynomially solvable problem, it loses its nice structure in general and for the problems considered in this work lead to computationally intensive solution methods.

We first present fast algorithms to project real transit system location data on road networks to obtain accurate estimations of travel time probability distributions. We then use decomposition techniques that improve the solution time of risk averse problems in general and present combinatorial procedures to reduce the number of binary variables for a family of combinatorial network problems. In the last chapter, we study a class of Stackelberg security games where not properly modeling uncertainty can have the same disastrous consequences as in emergency dispatching. In the classic setup, a defender aims to maximize its utility covering a set of targets against not perfectly rational human adversaries. We extend this expected payoff maximization model into its risk averse minimization counterpart and consider the possibility of several types of attacker.

1.2 Contributions and outline

In chapter 2 we propose efficient methodologies to obtain accurate travel time distribution data and then present in chapter 3 algorithms to efficiently solve risk averse shortest path problems. In chapter 4 we propose reformulations and branching schemes for hard network problems and we finish this document with chapter 5 where we propose solution approaches to Stackelberg security games with a risk averse objective facing quantal response adversaries.

Distribution estimation via data projection This chapter gives an overview of existing GPS data projection and defines two new algorithms: One allowing optimal data projection with respect to a new distance criterion, and a fast heuristic method that enables on the fly projection for large datasets. We present computational results on real and generated networks. This chapter is an extension and revised version of the working paper [22]

Risk averse routing In this chapter we develop efficient algorithms to solve risk averse shortest path problems via sample average approximation, linearization techniques and decomposition methods. We considered the Entropic risk measure and Conditional value at risk in presence of correlations between the random variables. We present promising computational results on medium sized generated instances. This chapter is an extension and revised version of the working paper [23].

Reformulations for hard network problems In this chapter, we show how to model certain difficult network problems with fewer integer variables. We then derive constraint branching schemes taking advantage of the special structure of these new models. We present several approaches to apply our methodology to the *st*-path and the Hamiltonian circuit sets

with a special emphasis on the *st*-path and *st*-cut sets in DAGs. This chapter is preliminary work in the development of practical resolution schemes for hard network problems. This chapter is part of the working paper [20].

Risk averse Stackelberg games with quantal response adversaries In this chapter we present a novel approach to Stackelberg security games with human adversaries. We show a fast way to compute the best defense strategy when the defender is risk averse and propose a Nonlinear Integer Programming formulation to solve the problem when extended to several types of adversaries. This chapter is part of the working paper [24]

Chapter 2

Distribution estimation via data projection

2.1 Introduction

One of our objectives was the development of efficient and reliable algorithms for the problem of projecting GPS trajectories onto a graph representing a transportation network. These algorithms can enable estimates of travel conditions (say travel times) from the large potential sources of real-time GPS positioning data in a transportation system, such as cell phones (think waze), GPS on transit system vehicles, and pilot vehicles. Building travel time distributions requires repeatedly projecting a massive amount of travel information onto a graph representing the transportation network. It becomes therefore important to be able to reliably and efficiently project real travel information onto graphs.

Finding the correct path that generates a GPS trajectory is, however, more complicated than it appears at first sight. In addition to difficulties induced by the size of realistic transportation networks and the volume of GPS data to process - as part of an online application or to estimate accurately a distribution - the problem might not be well posed. Specifically, errors in the GPS data, the sampling rate of this data, and inaccuracies incurred in representing a transportation network can make it difficult to discern which is the road segment that corresponds to certain GPS trajectory. Therefore, an important quality measure of a solution method is how sensitive it is to the uncertainties in the information being processed. Previous work on projecting GPS data on a graph have used a wide range of methodologies: some designed algorithms locally fitting the data to a subgraph of the network that is closest to every measurement point ([84]). Namely, for each point of the GPS trajectory, they find the closest node or edge in the network, and return this possibly disconnected subgraph. The approach used a heuristic to decide how to identify the subgraph at the intersections, where it is most difficult to match. Other authors preprocess the input data with Kalman filters to eliminate inconsistent data, and then apply an algorithm that builds sequentially the output path ([89]). Some algorithms build the solution path sequentially based on geometric considerations and store a buffer of past states. This enables to backtrack if the current path

deviates too much from the trajectory ([17, 56]). The method introduced in [19] finds first paths that locally match the input GPS trajectory according to the Frechet distance, i.e.: whose local point to curve distance is minimum, and then stitches these partial solutions together. In [26] the authors simplify dense GPS trajectories with Douglas-Peucker based algorithms ([31]) and then identify the road segments that match these simplified trajectories locally. [49] introduced a similar approach applying linear regressions to simplify and depurate high rate sampling GPS trajectories. [66] used a fuzzy logic based algorithm to sequentially evaluate the direction to take when dealing with an intersection. The subgraphs found by these so-called point-to-point or point-to-curve matching methods are not necessarily connected, which limits their use when dealing with complex trajectories. The articles by [55] and [61] designed algorithms that find the most probable road taken by a trajectory assuming that the measurement error of the velocity data follows a normal distribution. To find a path - eventually containing loops - in the graph that matches the GPS trajectory, these prior works construct an acyclic network from the trajectory information and the network topology. The longest path on this related network is the path that maximizes the likelihood of generating that trajectory data. Our approach to project GPS trajectory data on a map is similar to this last idea, in that we also use a related network built from the GPS trajectory information and the topological information of the network. In our work we take into account the geometrical considerations between the GPS trajectory and the network topology and also some upper bound of the maximum measurement error of modern GPS devices. We propose two optimization-based algorithms to find the path of graph edges that correspond to a given sequence of geo-referenced points. A driving feature of our approaches is that they lead to computationally efficient methods as we apply these algorithms to large amounts of data or looking for an immediate response. The first method modifies the arc traversal cost in proportion with how close the geo-referenced points are, thus making the shortest path between the first and last geo-referenced points on this modified network a good candidate for the path that generated these points. As for previous methods in the literature, this approach is sensitive to the density of GPS points and their measurement error hence leading to potentially mismatched paths. This motivates our second approach which, for new suitable definitions of the error incurred in selecting a certain path, selects the path that minimizes this error on a modified network similar to the one mentioned in [55] and [61]. It consists in measuring the error in a way that forbids to associate GPS points to edges of the graph that are not in the continuity of the path already constructed. Our computational results, on two networks that represent real road networks (Santiago, Chile and Seattle, WA) and a square grid, compare the accuracy and computational efficiency of the proposed methods. The networks we considered have several hundreds of thousands nodes and edges. We show that solving for the path that minimizes the error measure achieves the most accurate solutions, while the first approach achieves solutions that are slightly less precise but in a fraction of the running time in some cases.

Through this chapter we will use the following notations: We will consider that the transportation network is represented by a graph $G = (V, E)$, where the set of nodes has $|V| = n$ elements, and there are $|E| = m$ elements in the set of edges. We denote by d_e the non-negative length of edge $e \in E$, with $d \in \mathbb{R}^m_+$ its vector notation, and by $\{p_k\}_{k\in\{0,\dots,q\}}$ the sorted sequence of q trajectory points. By sorted we refer to having p_0 denote the first trajectory point and p_q the last one. We denote the non-negative distance of a given trajectory point p_k to node $v \in V$ by $d(p_k, v)$ and to edge $e \in E$ by $d(p_k, e)$.

We structured the rest of the chapter as follows: In the next section we present the first optimization based approach, which we refer to as Map Matching Heuristic (MMH). Section 2.3 introduces our second optimization based approach, referred to as Minimization of Oriented Error (MOE). We describe the computational experiments and their corresponding results in section 2.4. We present our conclusions in section 2.5.

2.2 Map Matching Heuristic (MMH)

In this section we present a heuristic method to select a path within the graph representing the transportation network that is close to the sequence of geo-referenced points, or trajectory of points for short. The central idea of this approach is to modify the graph by lowering the physical length of edges that are close to the trajectory of points. On this modified graph then solve a shortest path problem between the first point of the trajectory and the last one. This method guarantees that the subgraph returned is a path in the given network, which can be false with some of the existing projection methods. In addition to presenting the heuristic, in this section we describe algorithmic improvements, discuss its computational complexity and exhibit some examples where this heuristic has difficulty and fails to identify the correct path.

2.2.1 Heuristic Outline

The central idea of the heuristic is that for every point in the trajectory, say p_k, we find the edges of the graph that are within a radius $R > 0$ of p_k. We then lower their lengths by the distance $d(p_k, p_{k+1})$ between p_k and p_{k+1}. Once we have iterated this procedure over the entire sequence of trajectory points, we need to identify the starting and ending nodes according to the trajectory. For this we define two sets of candidate nodes

$$S := \{v \in V : d(p_0, v) \leqslant R\} \qquad \text{and} \qquad T := \{v \in V : d(p_q, v) \leqslant R\} \ .$$

The set S corresponds to nodes that are close to the first trajectory point p_0 and T to nodes that are close to the last trajectory node p_q.

As illustrated in figure 2.1 we define an artificial starting node s and connect it to every node $v \in S$ with a directed arc of length $d(p_0, v)$ and connect every node $v \in T$ to an artificial end node t with a directed arc of length $d(p_q, v)$. Since the nodes in S and T are closer than R to a trajectory point, the first part of this heuristic modified the edges incident to these nodes s and t. Since this procedure can potentially decrease the length of an arc multiple times, we make sure to keep the positive part of the modified edge lengths to avoid defining negative cost arcs. We select a path corresponding to a given trajectory of points from this modified network by solving a shortest path problem between s and t. Since all arcs are non-negative we can find the shortest path using Dijkstra's algorithm.

We note that if a trajectory point is close to an intersection, this procedure can reduce the length of all edges on the intersection an amount equal to the distance to the next trajectory

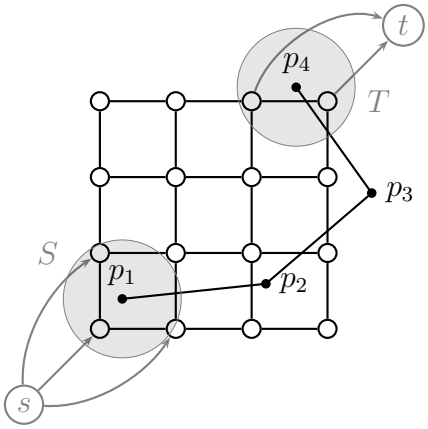


Figure 2.1: S/T zones.

point. This can reduce significantly the length of edges that are not oriented in the direction of the trajectory. For this reason it is preferable to reduce several times the length of edges with small quantities - due to several close trajectory points - rather than just one large reduction. This was achieved adding a number of artificial points in a straight line between every two consecutive trajectory points. We implemented this *densification* of the trajectory points imposing that the distance between consecutive points has to be at most d_{max}. This ensures that weight reductions of arcs that do not follow the trajectory are maintained small. We present these ideas in the pseudo-code of the MMH Heuristic outline in Algorithm 1 where we denote by `dijkstra`(G, w, u, v) the implementation of a Dijkstra algorithm that computes a shortest path inside graph $G = (V, E)$ with weights $w \in \mathbb{R}^m_+$ starting at node u and ending at node v.

Algorithm 1: Pseudocode of MMH Heuristic

Data: $G = (V, E)$, $P = (p_k)_{k\in\{1,\dots,q\}}$, R, $d \in \mathbb{R}^m$, $d_{\max}$
Result: A path SP

$w \leftarrow d$, $p_t \leftarrow p_1$;
for $k = 1, \dots, q-1$ **do**
  $n_k \leftarrow \lfloor d(p_k, p_{k+1})/d_{\max} \rfloor$;
  for $l = 0, \dots, n_k$ **do**
    $p_h \leftarrow p_k + (p_{k+1} - p_k) \cdot l \frac{d_{\max}}{d(p_k, p_{k+1})}$;
    for $e : d(p_t, e) \leqslant R$ **do** $w_e \leftarrow [w_e - d(p_t, p_h)]_+$;
    $p_t \leftarrow p_h$;
for $e : d(p_t, e) \leqslant R$ **do** $w_e \leftarrow [w_e - d(p_t, p_q)]_+$;
$S \leftarrow \{v \in V : d(p_0, v) \leqslant R\}$;
$T \leftarrow \{v \in V : d(p_q, v) \leqslant R\}$;
$\bar{G} = (V \cup \{s, t\}, E \cup \{(s, v)_{v\in S}, (v, t)_{v\in T}\})$;
for $v \in S$ **do** $w_{(s,v)} \leftarrow d(p_0, v)$;
for $v \in T$ **do** $w_{(v,t)} \leftarrow d(p_q, v)$;
$SP \leftarrow$`dijkstra`$(\bar{G}, w, s, t)$;
return SP

2.2.2 Complexity

There are at most

$$Q = \frac{1}{d_{max}} \sum_{k=1}^{q-1} d\left(p_k, p_{k+1}\right)$$

points belonging to the dense data. The complexity of the entire projection framework is as follows: for each of the (at most) Q points of the dense trajectory, find its closest edges has $O(m)$ complexity, giving a total complexity for all the dense points of $O(Qm)$. Dijkstra's algorithm using binary heap over the graph with modified weights has a complexity in $O(m + n \log n)$. Putting everything together, the total complexity of the projection algorithm for each trajectory is in $O(Qm + n \log n)$ which is fast in an algorithmic sense, but can be slow in practice because it requires the computation of the distance between every point-edge pair of the graph. To avoid to compute every combination, we propose in the following section a way to speed up the last method.

2.2.3 Improved version

First, we notice that the only weights we need during the execution of our algorithm are the ones picked by Dijkstra's algorithm. Consequently, when considering the current node with the lowest label, we only have to compute the modified weights w_e of its outgoing edges e. Although the theoretical worst-case complexity stays the same, doing so granted considerable speed-ups in practice. We present the pseudo-code of this Map Matching Heuristic (MMH) in Algorithm 2 where $(\bar{p}_k)_{k\in\{1,\dots,Q\}}$ represents the densified trajectory, H is a binary heap, `insert_heap(`H`,`v`,`π`)` is a routine that inserts the node index v with value π into the heap H, `change_val(`H`,`v`,`π`)` is a routine that changes the value of the node index v inside the heap H to the value π, `get_root(`H`)` is a routine that returns the node index of the root node of the heap H and `delete_root(`H`)` is a routine that deletes the root node of heap H.

During computational experimentation we observed that several paths had equal objective values, mainly because of entire zero-cost sections. To differentiate them, we decided to use a modified heap during the execution of Dijkstra's algorithm. This special heap follows a lexicographic order in terms of error first and then in terms of physical distance. This way, we still find a path with minimum cost with the extra requirement that amongst all the paths having the same cost, it finds the one having the shortest length. This modifications permits to tackle 'zero-zero' ties between edges.

2.2.4 Known issues

Some problems can show up with this heuristic: in some situations the algorithm can identify a path different from the original itinerary. One of these problems occurs when the trajectory passes through a dense street zone and has longer segments than the edges in the neighborhood. In this context the algorithm can choose to pass through closer but incorrect edges: as

Algorithm 2: Pseudocode of MMH heuristic on the fly

Data: $G = (V, E)$, $P = (\bar{p}_k)_{k\in\{1,\dots,Q\}}$, R, $d \in \mathbb{R}^E$, $d_{\max}$
Result: The last node of a path, a precedence array father$_i$ and distance labels π_i

```
w ← d;
T ← {v ∈ V : d(p_q, v) ⩽ R};
H ← ∅;
for v ∈ V do if d(p_0, v) ⩽ R then insert_heap(H, v, d(p_0, v));
while H ≠ ∅ do
    i ← get_root(H);
    delete_root(H);
    if i ∈ T then return i;
    for j ∈ δ⁺(i) do
        for k = 1, ..., Q − 1 do if d(p̄_k, (i, j)) ⩽ R then w_ij ← [w_ij − d(p_k, p_{k+1})]_+;
        if π_j > π_i + w_ij then
            π_j ← π_i + w_ij;
            father_j ← i;
            if j ∉ H then insert_heap(H, j, π_j) else change_val(H, j, π_j);
return T not reached.
```

we see in figure 2.2 where the dotted path is the real one and the dashed path the subgraph found by the algorithm. Note that this data-induced problem comes from low GPS sampling rates and is sometimes unavoidable with geometrical considerations only. The computational results we obtained on the grid network illustrate perfectly this bad behavior.

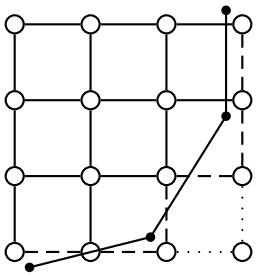


Figure 2.2: Sparse trajectory Vs. dense graph.

Another common partial failure of the algorithm can occur when the trajectory contains a tight U-turn like part. Due to the proximity between the forward and the backward pieces of the trajectory, the edges connecting them in the entire graph will have their lengths at low values. Sometimes those connecting edges will have lower values than the real path edges and consequently, as we can see in figure 2.3 the algorithm will cut off an entire U-turn part (dotted) passing directly at the end of it (dashed).

Although MMH is assured to have a short execution time, we have no guarantee to obtain a good solution. Moreover, the latter category of error is sometimes unavoidable and MMH

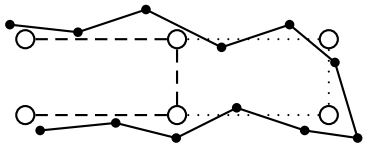


Figure 2.3: Tight U-turn type errors.

cannot identify path containing cycles or backtracking parts. In the next section we present a way to avoid the latter category of error and have a guarantee that we find the closest path with respect to some special error measure.

2.3 Minimum Oriented Error algorithm (MOE)

2.3.1 Oriented error measure

One of the first and simplest way to find a map matching was to associate each point to its closest edge in the network. One of the drawbacks of this method was to have total freedom over the choice of the edge for each point. For example, in figure 2.4 we can see that point p_5 is associated to an edge that cannot possibly be the right one.

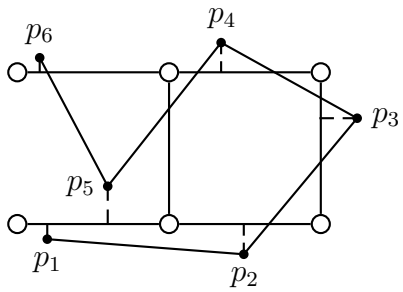


Figure 2.4: Classical closest point-edge association.

This type of wrong association appears because we allow the projection on any edge without any consideration for the topology of the network. From this observation we introduced the notion of *oriented error* of a subgraph. As for the classical error, the *oriented error* is a measure that computes the closest edge of each trajectory point with the important difference that it allows a point-edge association only if this edge is either the edge associated to the last point or one of its outgoing edges. We show an example of such a situation in figure 2.5.

Formally, given some path $C = \left(c_1, ..., c_{|C|}\right) \subseteq E$ we define its ordered error as follows:

$$OE(C) := \frac{1}{q} \sum_{k=1}^{q} d(p_k, c_{e(k)})$$

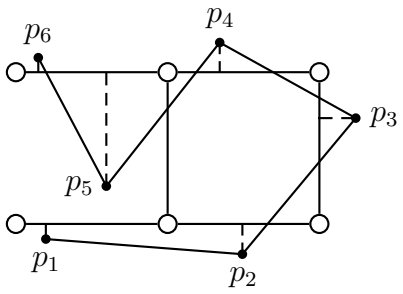


Figure 2.5: Oriented closest point-edge association.

With:

$$\begin{cases} e(k) := \arg\min \{d(p_k, c_i) : i \geqslant e(k-1)\} \\ e(0) := 1 \end{cases}$$

It represents the least distance between a point and the edges of the path C following the edge associated with the previous point. We can notice that this error measure heavily penalizes the paths cutting U-turns. In the following section we present an algorithmically fast way to find the optimal path for the ordered error measure.

2.3.2 Shortest path reduction

To find a path that minimizes the oriented error, we only have to find a minimum oriented error point-edge association such that any point is projected on the edge associated with the previous point or one of its outgoing edges. We will show that this procedure yields a polynomial time complexity using a shortest path algorithm in a derived network. Define the directed graph $\bar{G} = (\bar{V}, \bar{E})$ such that $\bar{V} = \{s\} \cup V_1 \cup V_2 \cup ... \cup V_q \cup \{t\}$ where for each point p_r of the trajectory we define the node set V_r as $V_r := \{v_r^e, \forall e \in E\}$. We define the edge set $\bar{E}$ as the union $\bar{E} = E_1 \cup E_2 \cup ... \cup E_q \cup E_{q+1}$ where the subsets E_q are defined as follows:

$$\begin{cases} E_1 = \{(s, v_1^e), \forall e \in E\} \\ E_r = \bigcup\limits_{(i,j)\in E} \left\{ (v_{r-1}^{(i,j)}, v_r^{(i,j)}), \left((v_{r-1}^{(i,j)}, v_r^{(j,k)}) \right)_{k \in \delta^-(j)} \right\} \quad \forall r \in \{2, ..., q\} \\ E_{q+1} = \left\{ (v_q^e, t), \forall e \in E \right\} \end{cases}$$

With the following edge weights:

$$\begin{cases} w_{(s, v_1^e)} = d_{p_1}^e & \forall e \in E \\ w_{(v_{r-1}^{(i,j)}, v_r^{(i,j)})} = d_{p_r}^{(i,j)} & \forall r \in \{2, ..., q\}, \forall (i,j) \in E \\ w_{(v_{r-1}^{(i,j)}, v_r^{(j,k)})} = d_{p_r}^{(j,k)} & \forall r \in \{2, ..., q\}, \forall (i,j) \in E, \forall k \in \delta^-(j) \\ w_{(v_q^e, t)} = 0 & \forall e \in E \end{cases}$$

Where $d_p^e = d(e, p)$ is the distance between some edge e and some trajectory point p. In the following we show an example of graph G in figure 2.6 and its associated graph $\bar{G}$ in figure 2.7. Let us prove that we can extract a path that minimizes the total ordered distance with

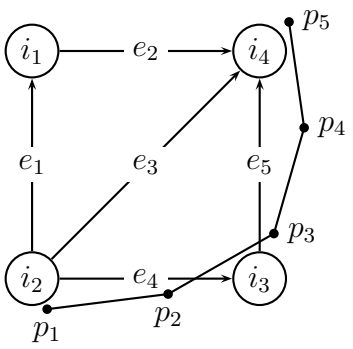


Figure 2.6: Example of graph with trajectory

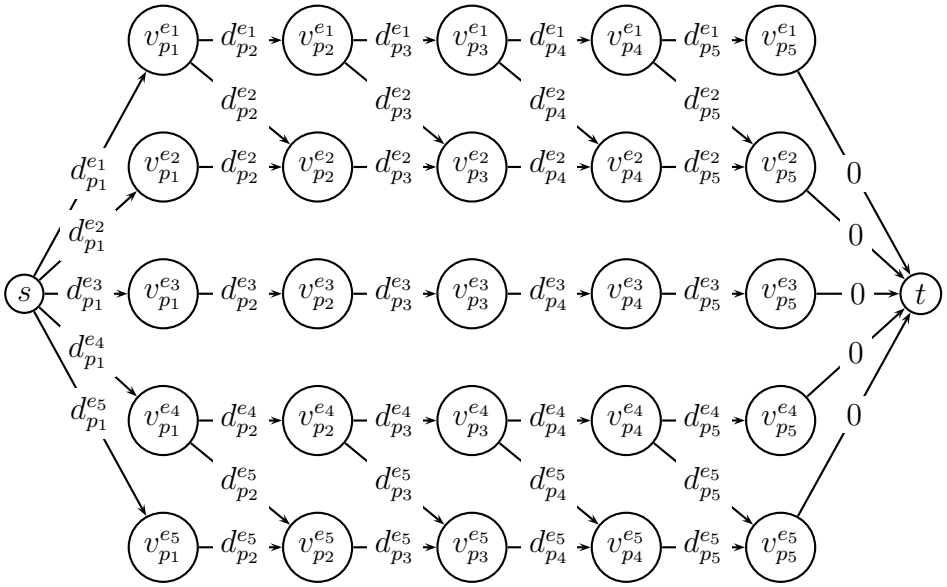


Figure 2.7: $\bar{G}$ associated to G.

the trajectory from every shortest (s,t)-path $C = (c_k)_{k\in\{1,...,q\}}$ in the graph $\bar{G}$. First, we can see that a (s,t)-path in $\bar{G}$ has exactly $q+1$ edges, the last one being dummy. The k-th edge c_k associates the k-th trajectory point p_k with some edge e_k of the graph and induces a cost equal to the distance between p_k and e_k. By construction, we know that e_k is an outgoing edge of e_{k-1} or e_{k-1} itself. Consequently, $(e_1, ..., e_k)$ defines a path in the original network G, and allows an association with the outgoing edges of e_{k-1} only. Putting everything together, a shortest (s,t)-path in $\bar{G}$ is giving an association path-trajectory that has the minimum oriented error. An important observation about this algorithm is enables to backtrack and form cycles, which is a good feature when we know that the trajectory loops at some point. Nevertheless, in this current form, it can lead to situations where, for instance, consecutive GPS points are successively projected on an edge and its counterpart. In the example of figure 2.8, we show that the path going through the edge $\{(i_1, i_2)\}$ and the path going through the edges $\{(i_1, i_2), (i_2, i_1), (i_1, i_2)\}$ have exactly the same oriented error. We tackled this problem using a double priority shortest path algorithm optimizing first with the oriented error, and when facing a tie between edges, take the one adding the least real distance to the actual path. In this goal, when running Dijkstra's algorithm, we used a double priority heap instead of the classical one: we first optimize with respect to the lengths of the network $\bar{G}$, and then we force it to choose the path with the minimum real path length.

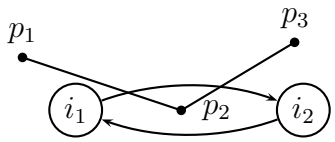


Figure 2.8: Example of potential auto-looping instance

2.3.3 Theoretical complexity

Noticing that $\bar{G}$ is acyclic, we can find a (s,t)-shortest path in $O(|\bar{E}|)$ time with a breadth first search or a depth first search. Let us compute the total complexity in function of the original parameters. The number of edges of each subset E_q is $|E_1| = |E_{q+1}| = m$ for the first and the $q+1$-th one, and we have $|E_r| = \sum_{(i,j)\in E} [1 + |\delta^-(j)|] = m + \sum_{i\in V} \delta^+(i)\delta^-(i), \forall r \in \{2, ..., q\}$ in between. Putting everything together the total number of edges in $\bar{E}$ is

$$\begin{aligned}
|\bar{E}| &= 2m + (q-1)\left(m + \sum_{i\in V} \delta^+(i)\delta^-(i)\right) \\
&= (q+1)m + (q-1)\sum_{i\in V} \delta^+(i)\delta^-(i) \\
&\leqslant m(q+1) + m(q-1)\min\left\{D_G^+, D_G^-\right\} = O\left(qm^2\right)
\end{aligned}$$

With $D_G^+ = \max_{i\in V} \delta^+(i)$ and $D_G^- = \max_{i\in V} \delta^-(i)$ respectively the in-degree and out-degree of G. Therefore the total complexity $\mathcal{C}(\bar{G})$ of finding a shortest (s,t)-path over graph $\bar{G}$ is $\mathcal{C}(\bar{G}) = O\left(qm^2\right)$.

2.3.4 An improved optimal algorithm

The computationally heavy part of the previous algorithm is the construction of the derived network $\bar{G}$. If we have some upper bound δ over the maximum error between the real path and its trajectory - a distance - we can omit *a priori* the point-edge associations having a distance greater than δ We give an example of such a situation in figure 2.9. This process permits to build the derived network $\bar{G}$ only with the edges of E in a corridor of width δ around the trajectory as depicted in figure 2.10. Further, we can use the same trick we used to improve the last heuristic to speed up the algorithm. Indeed, the only weights we need during the execution of Dijkstra's algorithm are the outcoming edges of the minimum label nodes. Consequently we can modify the shortest path algorithm in the same way and compute dynamically the distances between trajectory points and edges.

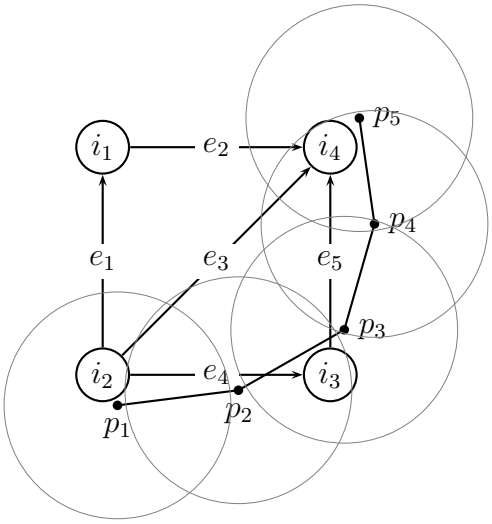


Figure 2.9: Instance with preprocessing.

2.4 Experimental results

2.4.1 Data sets

We ran our two algorithms over two real networks: Santiago de Chile (SAN, $n \approx 330000$ and $m \approx 660000$) and Seattle, WA (SEA, $n \approx 420000$ and $m \approx 860000$) and a ficticial grid network (GRI) of 100×100 nodes with edges of length 100 meters. For each network we generated 100 different paths between nodes distant of a distance between 5 and 10 kilometers (then generating path of at least that length). The generated paths are shortest paths between two nodes uniformly drawn in a box of the densest zone of its network with respect to a random uniform weight. Consequently, the family of trajectories we generated in this chapter have a minimal number of edges on average. Second, we sampled and noised with Gaussian perturbation each of these paths with a sampling rate in $s \in \{10, 30, 50, 70, 90, 110, 130, 150, 170, 190, 210, 250, 300, 400, 500\}[m]$ and a Gaussian error $e \sim \mathcal{N}(\mu = 0, \sigma)$ such as $\sigma \in \{1, 2, 5, 7, 10, 20, 50\}$[m] in both directions, without correlation between them. The total number of instances we used was then 31500 (3[networks] x 100[trajectories] x 15[s] x 7[σ]). The parameters we used for the algorithms were the following: two consecutive points of the trajectory must be closer than $d_{max} = 20[m]$ and the radius parameter used in the algorithms is $\delta = R = 500[m]$

We had about 900 real GPS trajectories for the particular case of Santiago de Chile, and applied our two methods on them without knowing what their real paths were. This implied that we were unable to compute the real difference between the original path and the output of the algorithms. For these real instances we only present the oriented error and classic path to trajectory distance.

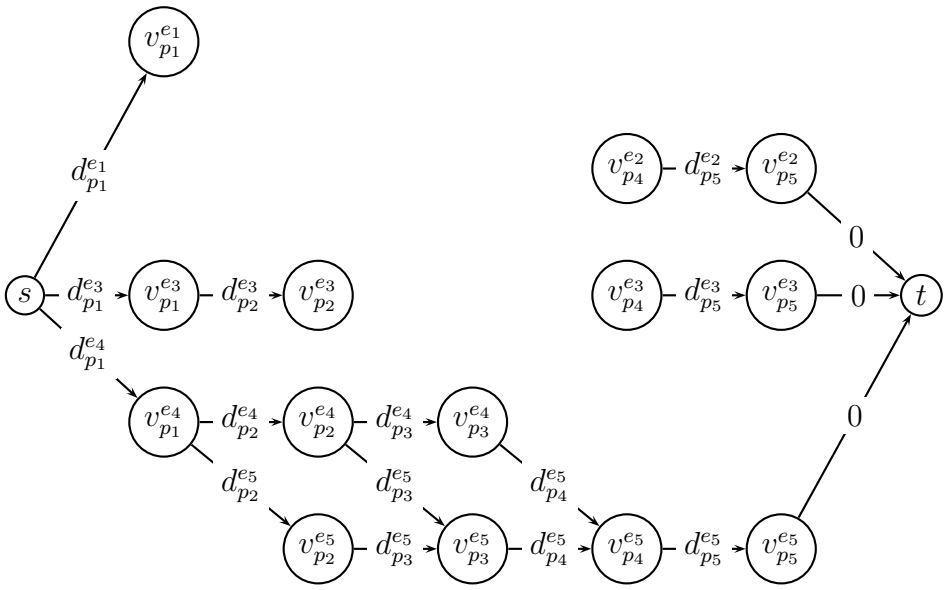


Figure 2.10: Preprocessed $\bar{G}$.

2.4.2 Results

Define $R = \left(r_1, ..., r_{|R|}\right) \subseteq E$ to be the real path of the trajectory. To check if the algorithm is efficiently identifying the correct subgraph of each trajectory, we defined an error measure comparing the real path R and the one computed by some algorithm C. Define the distance $d(e, a)$ between two edges $e = (i, j) \in E$ and $a = (u, v) \in E$ as the minimum distance between two points of the segments defined by the two edges. In other words:

$$d(e, a) := \min_{(\alpha,\beta)\in[0,1]^2} d\left(\alpha i + (1-\alpha)j, \beta u + (1-\beta)v\right)$$

assuming the abuse of notation that a node v can represent its geometric point as well. Now we can define the notion of maximum real error of path C, $RE(C)$:

$$RE(C) := \max_{i\in\{1,...,|C|\}} \min_{j\in\{1,...,|R|\}} d(c_i, r_j)$$

that represents the maximum distance between an edge of the path found and the real path the trajectory comes from. The classic error measure, defined as the average of the point-to-closest-edge distances, is written as follows:

$$CE(C) := \frac{1}{q} \sum_{k=1}^{q} \min_{i\in\{1,...,|C|\}} d(p_k, c_i)$$

The algorithms presented in this chapter were coded in C programming language and run over a cluster node of 2.4GHz with 6Go Ram.

In Table 2.1 we will compare the real error (RE), the oriented error (OE) and the classic error (CE) of the solutions returned by MMH and MOE ever the networks we considered.

The numbers presented in table 2.1 are the average and maximum from the 10500 instances generated for each network (i.e.: 31500 for the overall line total). As expected, all the types

Table 2.1: General statistics

Network	Algorithm	Geom. avg $T[s]$	$RE(C)[m]$ max	$RE(C)[m]$ avg	$OE(C)[m]$ max	$OE(C)[m]$ avg	$CE(C)[m]$ max	$CE(C)[m]$ avg
SAN	MOE	2.72	13.26	0.59	5.28	0.86	3.94	0.73
	MMH	2.08	15.24	0.91	7.57	1.10	6.97	1.04
SEA	MOE	2.64	14.63	0.63	4.12	0.71	3.57	0.66
	MMH	2.04	16.60	1.20	8.11	1.30	7.49	1.24
GRI	MOE	2.80	18.77	0.51	3.01	0.72	2.54	0.67
	MMH	1.91	15.56	1.19	6.64	1.11	6.56	1.08
Overall	MOE	2.72	15.56	0.57	4.14	0.76	3.35	0.69
	MMH	2.01	15.80	1.10	7.44	1.17	7.01	1.12

of error measures are lower for the paths found by the optimal algorithm MOE than those computed by the heuristic MMH. In particular, this result holds for the real error measure, meaning that the map matching of the optimal algorithm is better than for the heuristic but also takes up to 35% of extra computation time on average. Figure 2.11 show that the real error measures of both algorithm depend of the sampling rate in an almost linear way ($RE(C) \approx 0.07 \times s$). Again, we can see that MOE behaves always better than MMH on average.

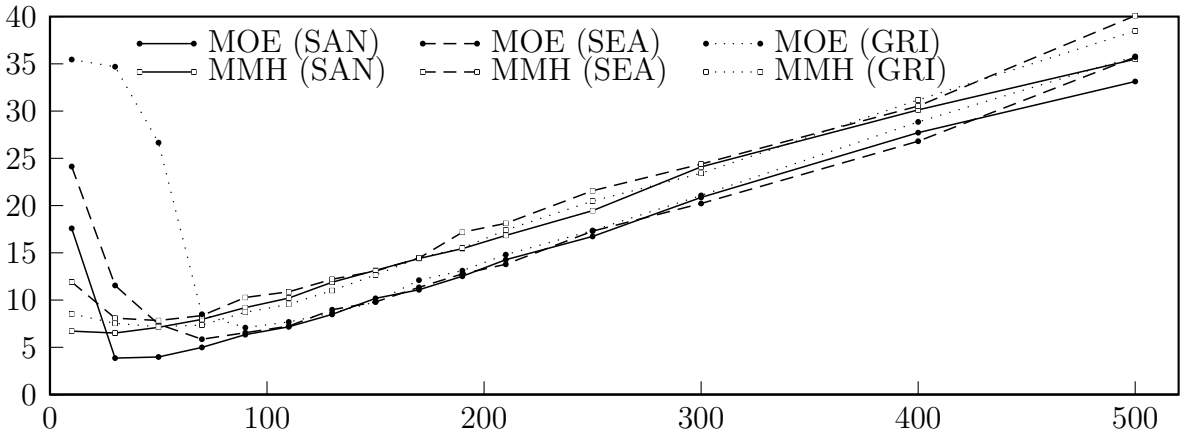


Figure 2.11: $\mathbb{E}\left[RE(C)\right][m]$ Vs. Sampling step $[m]$

Furthermore, figure 2.12 shows that the standard deviation of the Gaussian noise has close to no influence up to $\sigma \leqslant 20[m]$, which is in the range of modern GPS sensitivity. In figure 2.13 we can see that the sampling step has a weak influence on the execution time of both algorithms given that our densification procedure always put back to instances where the artificial sampling step is at most d_{max}. Nevertheless, its influence comes from the presence of similarly good candidates for the edges to project the trajectory on. The presence of these candidates implies a deeper - then slower - search of Dijkstra's algorithm. In figure 2.14 we can observe that the standard deviation σ of the Gaussian noise we applied has an influence on the execution time for the MOE algorithm. Indeed, a high dispersion of the trajectory raises the size of the derived network $\bar{G}$ - i.e.: the number of possible paths to explore - then slowing the overall execution time of Dijkstra's algorithm. In figure 2.15 we can see that any

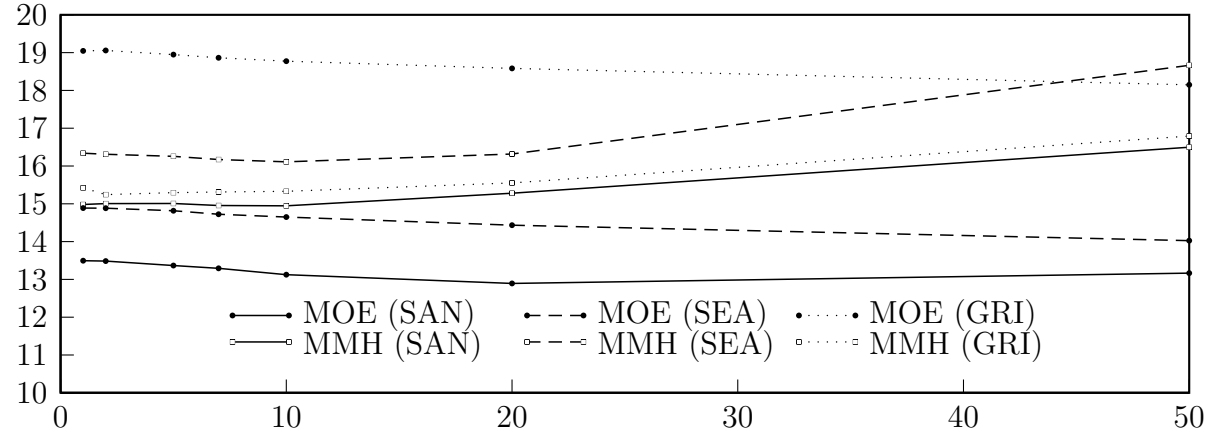


Figure 2.12: avg. $RE(C)[m]$ Vs. noise's standard deviation $\sigma[m]$

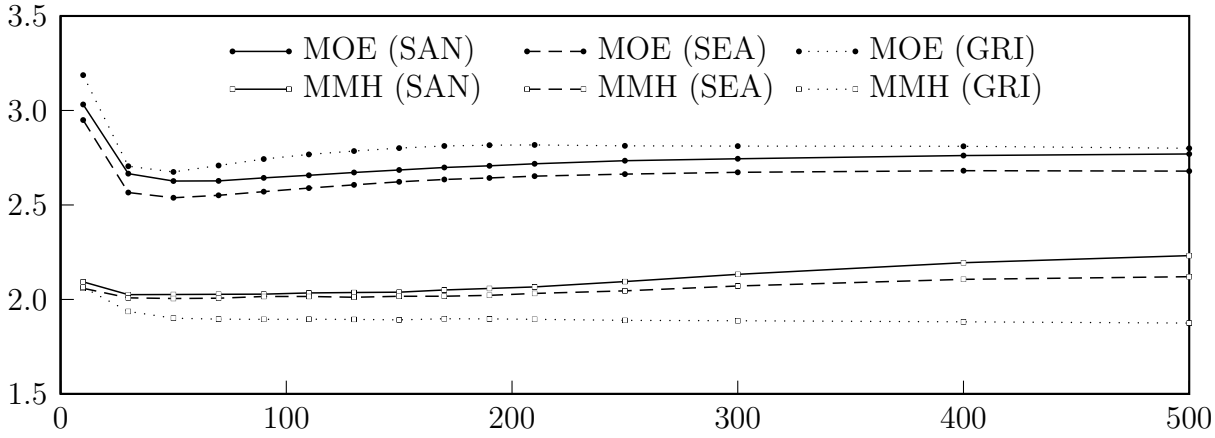


Figure 2.13: Geometric mean $T[s]$ Vs. Sampling step $[m]$

algorithm on any network (Santiago (SAN), Seattle(SEA), Grid(GRI)) returns (on average) map matching paths where at least 89% of their edges are within a corridor of 1 meter from the real path. Once again, we can see that MOE behaves better than MMH with at least 92% on average of their path edges in a corridor of one meter around the real path. Further, on the artificial grid instances, we can see that there is a peak of the cumulative distributions at 100 meters, which corresponds to the length of all the edges, and another one at 133 meters which roughly corresponds to $100\sqrt{2}$ meters, i.e.: to point-edge associations wrongly associated with a neighboring edge or an edge which is at an opposite corner (at a distances of $100\sqrt{2}$ meters) of the square the edge belongs to. The same phenomenon occurs as well at 200 and 233 meters.

Table 2.2: Transantiago statistics

		$OE(C)[m]$		$CE(C)[m]$	
Algorithm	Geometric mean $T[ms]$	max	avg	max	avg
MOE	3.21	137.68	12.53	65.09	3.29
MMH	0.08	317.71	30.82	303.56	26.52

We tested the algorithms on the real trajectories coming from the public transportation

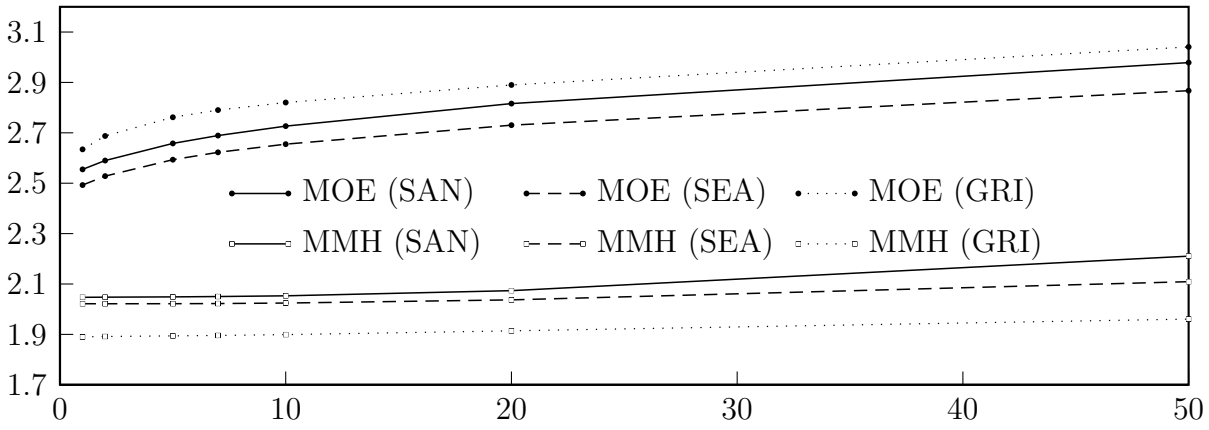


Figure 2.14: Geometric mean $T[ms]$ Vs. $\sigma[m]$

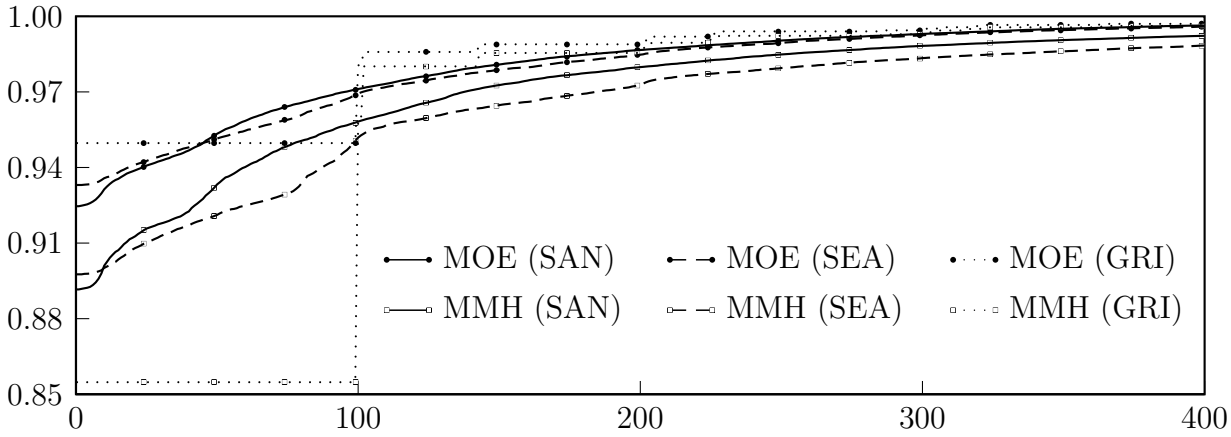


Figure 2.15: Cumulative distributions. Proportion of edges Vs. Distance to real path $[m]$

system of Santiago City. We summarize the general statistics over these real instances in table 2.2 Again, we see that MOE has a way better behavior than MMH in terms of distances between the output path and the trajectory. Indeed, on average, a trajectory point is four times closer of the output path returned by MOE than the path returned by MMH. On another hand, MMH is 35% faster than MOE on average.

2.5 Conclusions

In this chapter, we defined an oriented measure of error that takes in account the ordering of the path when computing its discrepancy level with the trajectory. We developed an algorithm that finds in an optimal way the path minimizing the oriented error measure and added some algorithmic steps speeding it up when we have information about the measurement error of the GPS signal. This algorithm can yield prohibitive execution times when used as a dynamic map matching routine projecting very large trajectories. To do so, we constructed a heuristic that runs on average 35% faster at a - relatively small - cost of the fidelity of the path it finds and the impossibility of detecting cycles or backtrack. The work in this chapter constitutes a way to develop accurate estimates of travel time distributions on a large portion of the different arcs representing the road network and thus to generate

a risk averse dispatch of emergency vehicles under different congestion conditions. Being able to construct a distribution of travel times on each arc becomes a first step to accurately evaluate different risk measures.

Chapter 3

Risk averse routing problems

3.1 Introduction

Real life problems such as the vehicle routing problem ([50]), the facility location problem ([33]) or the lot sizing problem ([32, 41]) contain inherent shortest path problems as an important component. Given that deterministic shortest path sub-problems are solvable in polynomial time ([3]), decomposition techniques are a natural option to tackle these problems in a fast way. Nevertheless, assuming deterministic data is unreasonable in a number of applications and doing so can lead to poor solutions in practice ([15]). Consequently, other ways to evaluate the cost of a path that take into account its uncertainty are necessary. Unfortunately, doing so spoils the special properties of the classical shortest path problem and leads to problems that are more difficult to solve. For instance, the shortest path problem satisfies total unimodularity, which guarantees that when we use a linear objective, the linear programming relaxation reaches an integer optimal solution. Nevertheless, taking into account uncertainty leads to a convex objective that cannot make use of this property. Previous work has put a lot of effort into methods to incorporate uncertainty in optimization problems ([10, 73, 83]). For an introduction to stochastic programming in general, the reader may refer to [16, 70] or [14] for an introduction to robust optimization. Specific methods have been developed for optimization problems under uncertainty exploiting particular problem structure. For instance, the lot sizing problem with uncertain costs or lead times in [8], the vehicle routing problem with time windows and uncertain data in [37] and the facility location problem with uncertainty in [71].

Taking the expected value of the cost of a path appears as a natural way to quantify the uncertain travel time of the path but ignores the dispersion of the underlying distributions, which can be significant. For example, consider two nodes s and t connected by two paths. The first path has a fixed travel time of 6 minutes and the second one has an uncertain travel time, it can be 1 minute or 9 minutes, each with probability 0.5. Then, in expected value, the best choice is the second path with an expected travel time of 4.5 minutes, but it has the highest variability between the two possibilities. Taking the second path is riskier with a 50% delay half of the time. This amount of delay can be unacceptable in certain applications, such as emergency dispatching systems, where increases in delay can lead to

increases in fatalities. This notion of risk aversion was mathematically defined in the early work of [80] where the risk perceived by a rational agent is modeled by the expected value of their convex disutility function. Such a way to define risk is also called certainty equivalent under some utility function ([87, 86]).

A risk measure $\rho(\cdot)$ is a real valued mapping of a set of random variables used to quantify the degree of risk aversion of a random variable. There are some desirable features a risk measure could have. It should be normalized, i.e.: the zero random variable has zero risk, invariant by translation, i.e.: when minimizing a random cost X, the risk associated to the random variable $X + a$ for any real a is exactly $\rho(X) + a$, and finally should be monotone, i.e.: if a random variable X dominates stochastically another random variable Y ([53]) which we write $X \succeq Y$, then we have $\rho(X) \leqslant \rho(Y)$ in the case of a minimization problem, and $\rho(X) \geqslant \rho(Y)$ in a maximization context. Convexity is also a desired property of a risk measure. For example, when optimizing a portfolio mixing different assets is a good way to reduce the dispersion of a portfolio in practice. The latter can be modeled via the convexity of the risk measure considered. The expected value or the CVaR of a random variable are typical examples of convex risk measures. The concept of coherent risk measures axiomatically defines the set of properties that a risk measure should satisfy ([6, 7]), and shows that CVaR is a coherent risk measure. In [48] they prove that CVaR is central among coherent risk measures, showing that any coherent, co-monotone and law invariant risk measure - or distortion risk measures - is equivalently representable as a convex combination of CVaRs at different levels of security.

An early work presenting a solution approach for a risk averse problem appears in [57], where they use the variance to represent the dispersion of the data, giving birth to the mean-risk model. [62, 54] used this latter model to solve CVaR minimization shortest path problems with uncorrelated and normal distributions on edge travel times. [67] proved that minimizing a CVaR objective is a standard convex optimization problem. Nevertheless, this requires the minimization of an expectation, which can even be difficult to compute for a large number of correlated random variables. Sample average approximation (SAA) methods ([44]) provide a standard framework to approximately solve these difficult problems. To the best of our knowledge, all the literature on risk averse shortest path problems assumes uncorrelated uncertain travel times (or costs). Ignoring the correlation that may be present in random travel times might return shortest paths solutions failing to consider the path travel time distribution ([2, 1]). As shown in [88] the correlation in uncertain travel times is indeed present in transportation networks.

In figures 3.2 and 3.1 we see an illustration of the influence of correlations on the shortest path problem. Let $X, Y, Z \sim \mathcal{B}(1, 3)$ be Bernoulli random variables that can take the values 1 or 3 with the same probability 0.5. The risk associated to a path P in this example will be the mean-risk objective, given by the sum of the expected value and variance over path P, explicitly $\rho(P) = \mathbb{E}[P] + \sigma^2[P]$. Let P_1 and P_2 be the paths passing respectively through the upper edge and the lower edge. In a first case (figure 3.1) there is no correlation between the three edges. We have $\mathbb{E}(P_1) = 4$, $\mathbb{E}(P_2) = 3$ and $\sigma^2(P_1) = \sigma^2(P_2) = 2$. The risk associated to the path P_1 is then $\rho(P_1) = 6$ while the risk of the lower path is $\rho(P_2) = 5$, making the second one preferable. In the second case (figure 3.2) the distributions remain the same, but we assume there is full correlation between the edges' travel times, i.e.: $X = Y = Z$. We

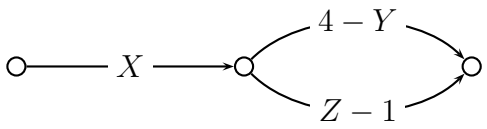


Figure 3.1: Without considering correlations

have $\mathbb{E}(P_1) = 4$, $\mathbb{E}(P_2) = 3$, $\sigma^2(P_1) = 0$ and $\sigma^2(P_2) = 4$. The risk associated to the path P_1 passing through the upper edge is $\rho(P_1) = 4$ while the risk of the lower path is $\rho(P_2) = 7$ meaning that the first one is chosen. In this example we can see that ignoring correlations

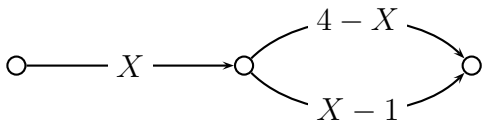


Figure 3.2: Considering correlations

can lead to take a path that is 75% worse than the optimal one. SAA methods can handle correlations between the random variables. Nevertheless, it may sometimes require large samples to guarantee the quality of the solution obtained ([54]) at the cost of solving larger optimization problems.

In this chapter, we assume a given directed network $G = (V, E)$ with $m = |E|$ edges, $n = |V|$ nodes, a source node $s \in V$, and a sink node $t \in V$. We define the set of paths from s to t on G with the standard st-path polytope ([3]) below

$$x \in X \;=\; \left\{ x \in \{0,1\}^m \;:\; \sum_{k:(j,k)\in E} x_{jk} - \sum_{i:(i,j)\in E} x_{ij} = b_j \quad \forall j \in V \right\} , \tag{3.1}$$

where the right hand side vector b satisfies $b_s = 1$, $b_t = -1$, and $b_j = 0$ for all $j \in V \setminus \{s, t\}$. The uncertain shortest path problem considers uncertain costs c_{ij} on every edge $(i, j) \in E$ such that the cost vector $c = (c_{ij})_{(i,j)\in E}$ follows some distribution that we can sample. We make no further assumptions regarding the uncertain cost vector, in particular we do not assume the uncertainty between arcs is independent.

Given an st-path solution $x \in X$ and an uncertain cost vector c, its total cost $c^\top x$ is a random variable that we can evaluate with a convex risk measure with $\rho\left(c^\top x\right)$. The problem considered in this work is to find the st-path which minimizes this risk aversion measure, namely

$$(P_\rho) \qquad\qquad w^* = \min_{x\in X} \rho\left(c^\top x\right) \;. \tag{3.2}$$

In this chapter, we will only consider risk measures of the form $\rho\left(c^\top x\right) = \mathbb{E}\left[u\left(c^\top x\right)\right]$ with u some positive, nondecreasing and strictly convex disutility function such that $\rho(0) = 0$. At least with this type of risk measures, we can prove that problem (3.2) is NP-hard by reducing it to the set partition problem: Given a set of N positive integers $(a_i)_{i\in\{1,\dots,N\}}$, we want to

find a partition $\{N_1, N_2\}$, $N_1 \cap N_2 = \emptyset$, $N_1 \cup N_2 = \{1, ..., N\}$ such that $\sum_{i \in N_1} a_i = \sum_{i \in N_2} a_i$. Assuming that the a_i are "deterministic random variables", finding the shortest path in the

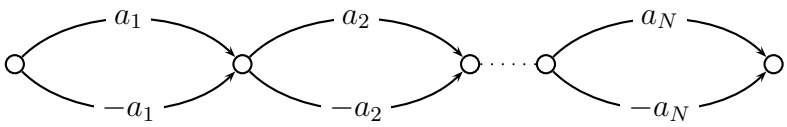


Figure 3.3: Reduction to Set Partition Problem

network depicted in figure 3.3 with respect to the quadratic risk measure $\left(c^\top x\right)^2$ solves the set partition problem. Indeed, for this construction we have that this shortest path problem has an optimal solution of value 0 if and only if the parameters $(a_i)_{i \in \{1,...,N\}}$ define a solvable instance of the partition problem. Further, if the instance is not solvable then the optimal path defines the partition minimizing the difference of weights between N_1 and N_2.

In general, even the evaluation $\rho\left(c^\top x\right)$ of some solution $x \in X$ can be challenging. We can approximate this optimization problem using SAA methods if we can sample the uncertain total cost. In this chapter we propose efficient implementations of the SAA methodology to solve large instances of the risk averse shortest path problem (P_ρ) in (3.2). In particular we focus on algorithms to solve problem (P_ρ) when using the Conditional Value at Risk measure and the Entropic risk measure.

In the next section we present basic definitions and the general framework of the SAA methodology. We present the different formulations and algorithms that we implemented to solve (P_ρ) under each risk measure in section 3.2. This section states the SAA problems that we have to solve and introduces the decomposition algorithms implemented for each risk measure. We discuss simplifications that are possible when considering the entropic risk measure with uncorrelated uncertainty. The chapter then presents the computational results in section 3.4 and concludes in section 3.5.

3.2 Solution Approaches Proposed

In the following, we assume that we can generate samples of the random variable c. We focus our attention on SAA methods to solve these problems. Given a sample of S realizations of c, $(c^s)_{s \in \{1,...,S\}}$ of respective probabilities $(p_s)_{s \in \{1,...,S\}}$, we can approximate the risk of a solution $x \in X$ as follows:

$$\mathbb{E}\left[u\left(c^\top x\right)\right] \approx \sum_{s=1}^{S} p_s u\left(\left(c^s\right)^\top x\right) \tag{3.3}$$

Consequently, the sample average approximation of (3.2) can be written:

$$(P_S) \quad w_S := \min_{x \in X} \left\{ w_S(x) := \sum_{s=1}^{S} p_s u\left(\left(c^s\right)^\top x\right) \right\} \tag{3.4}$$

In our study, we will consider two specific risk measures for the random cost $c^\top x$: the Conditional Value at Risk which for a given tolerance $\varepsilon \in]0,1]$ can be written as described in [67]:

$$\mathrm{CVaR}_\varepsilon(c^\top x) = \min_{z\in\mathbb{R}} \left\{ z + \varepsilon^{-1}\mathbb{E}\left[\left(c^\top x - z\right)_+\right]\right\} \tag{3.5}$$

and the Entropic Risk Measure ([65, 5]) at level $\alpha \in \mathbb{R}$ that can be defined as:

$$\mathrm{E}_\alpha\left(c^\top x\right) = \alpha \ln \mathbb{E}\left[e^{\frac{c^\top x}{\alpha}}\right] \tag{3.6}$$

Notice that the entropic risk measure is a risk aversion measure whenever $\alpha > 0$, a risk seeking measure when $\alpha < 0$ and models a risk neutrality as $\mathrm{E}_\alpha\left(c^\top x\right) \underset{\alpha\to 0}{\longrightarrow} \mathbb{E}\left[c^\top x\right]$. We remark that minimizing risk measure (3.6) is equivalent to minimizing $\mathbb{E}\left[e^{\frac{c^\top x}{\alpha}}\right]$. This allows us to use the SAA formulation (3.4) to solve the optimization problem. The CVaR risk measure is equivalent to the minimization problem (3.5). This shows that it is the minimization of the expectation of some function.

3.2.1 Conditional Value at Risk

Monolithic form (CM). A first way to solve the CVaR minimization problem is to directly solve its associated sampled problem (3.7):

$$(\mathrm{CVaR}_\varepsilon^S) \quad w_S := \min_{x\in X}\left\{ \mathrm{CVaR}_\varepsilon^S(c^\top x) := \min_{z\in\mathbb{R}} z + \varepsilon^{-1}\sum_{s=1}^S p_s\left(\left(c^s\right)^\top x - z\right)_+\right\} \tag{3.7}$$

which is equivalent to the following MIP formulation:

$$w_S := \min_{z,\eta,x} z + \varepsilon^{-1}\sum_{s=1}^S \eta_s \tag{3.8a}$$

$$(\mathrm{CVaR}_\varepsilon^S) \qquad \text{s.t. } : x \in X \tag{3.8b}$$

$$\eta_s \geqslant p_s\left(\left(c^s\right)^\top x - z\right) \qquad \forall s \in \{1,\dots S\} \tag{3.8c}$$

$$\eta \geqslant 0 \tag{3.8d}$$

Given some feasible solution $x \in X$, an important remark done in [35] provides a quick way to compute an estimated value $\mathrm{CVaR}_\varepsilon^S(c^\top x)$ of $\mathrm{CVaR}_\varepsilon(c^\top x)$. Indeed, given a sample $\left(c^s, p_s\right)_{s\in\{1,\dots,S\}}$ we have:

$$\mathrm{CVaR}_\varepsilon^S(c^\top x) := \min_{\eta,z}\left\{ z + \varepsilon^{-1}\sum_{s=1}^S p_s\eta_s : \eta_s \geqslant \left(c^s\right)^\top x - t, \eta_s \geqslant 0, t \in \mathbb{R}\right\} \tag{3.9}$$

when taking the dual and changing variables we obtain:

$$\mathrm{CVaR}_\varepsilon^S(c^\top x) := \max_{\pi}\left\{ \varepsilon^{-1}\sum_{s=1}^S p_s\pi_s\left(c^s\right)^\top x : \sum_{s=1}^S p_s\pi_s = \varepsilon, \pi \in [0,1]^S\right\} \tag{3.10}$$

We can solve this second problem sorting the scenarios by decreasing $c^s x$ and fill the equality constraint with the best scenarios. Let suppose we have such an ordering $c^{(1)}x \geqslant c^{(2)}x \geqslant ... \geqslant c^{(S)}$ and let $s^* := \max\left\{s' \in \{1,...,S\} : \sum_{s=1}^{s'} p_{(s)} \leqslant \varepsilon\right\}$. Then it is easy to see that an optimal solution for the dualized problem (3.10) is:

$$\pi_s^* = \begin{cases} 1 & \forall s \in \{(1),...,(s^*)\} \\ \varepsilon - \sum_{s'=1}^{s^*} p_{(s')} & s = (s^*+1) \\ 0 & \forall s \in \{(s^*+2),...,(S)\} \end{cases} \tag{3.11}$$

We can compute this optimal solution in $O(S \ln S)$ time and extract an upper bound on the optimal value of problem (3.8) from it. Based on the solution given by (3.11) we define the partition induced by x as:

$$\mathcal{Q}_x := \{\{(1),...,(s^*)\},\{(s^*+1)\},\{(s^*+2),...,(S)\}\} \tag{3.12}$$

Although we are able to cast the problem directly as a MIP and evaluate its estimated value quickly, we expect this monolithic form to be slow to solve in practice due to the large number of scenarios we must potentially generate to have a good approximation. To tackle this, we will present in the following two algorithms tailored to solve the monolithic formulation: a subgradient algorithm and a decomposition based method.

Subgradient (CS). As suggested in [47], given that the functions $\rho(c^\top \cdot) : x \longmapsto \rho\left(c^\top x\right)$ are convex, we can represent them as the point-wise maximum of its subgradients. Replacing in problem (3.2) we obtain formulation (3.13):

$$w^* := \min_{\eta,x} \quad \eta \tag{3.13a}$$

$$(\partial P_\rho) \quad \text{s.t. } : x \in X \tag{3.13b}$$

$$\eta \geqslant \rho\left(c^\top x^0\right) + d^\top (x - x^0) \qquad \forall x^0 \in X, \forall d \in \partial[\rho(c^\top \cdot)](x^0) \tag{3.13c}$$

Which has an infinite number of constraints in general. Nevertheless, the piecewise linearity of CVaR allows to cast the latter problem (3.13) into the following formulation (3.14):

$$w_S := \min_{z,x,\eta} \quad z + \varepsilon^{-1}\eta \tag{3.14a}$$

$$\left(\partial \text{CVaR}^{\mathcal{S}}\right) \quad \text{s.t. } : x \in X \tag{3.14b}$$

$$\eta \geqslant \sum_{s \in C} p_s \left(\left(c^s\right)^\top x - z\right) \qquad \forall C \in \mathcal{S} \tag{3.14c}$$

Where $\mathcal{S}$ is the set of all subsets of $\{1,...,S\}$. This suggests an iterative algorithm solving relaxed problems and generating cuts whenever constraints (3.14c) are violated. We first solve a relaxation of problem (3.14) containing only one constraint of type (3.14c) with $\mathcal{S} = \{\{1,...,S\}\}$. Let $(\widetilde{x}, \widetilde{z}, \widetilde{\eta})$ be the current optimal solution of problem (3.14) of objective value $\widetilde{w}$. We remark that $(\widetilde{x}, \widetilde{z})$ is feasible for (3.8), so we have $\widetilde{w} \leqslant \widetilde{z} + \varepsilon^{-1} \sum_{s=1}^{S} \left(\left(c^s\right)^\top \widetilde{x} - \widetilde{z}\right)_+$.

Let us define $\widetilde{C} = \left\{ s \in \{1, ..., S\} : (c^s)^\top \widetilde{x} - \widetilde{z} > 0 \right\}$. If $\widetilde{w} \geqslant \sum_{s \in \widetilde{C}} p_s \left[(c^s)^\top \widetilde{x} - \widetilde{z} \right]$ then we can easily see that the current solution $(\widetilde{x}, \widetilde{z})$ is feasible and optimal for the complete problem (3.7). In the other case we add the constraint corresponding to $\widetilde{C}$ and we solve again the resulting problem with $\mathcal{S} \leftarrow \mathcal{S} \cup \widetilde{C}$. We remark that the first step with $\mathcal{S} = \{\{1, ..., S\}\}$ is equivalent to solve the original problem (3.2) with $\rho \equiv \mathbb{E}$, which is as hard as solving the deterministic version. In our context of a shortest path problem, any existing polynomial algorithm like Dijkstra's solves it efficiently. We show a pseudo code of this method in Algorithm 3.

Algorithm 3: Pseudocode of CS algorithm

Data: A problem $(\text{CVaR}_\varepsilon^S)$
Result: A δ-optimal solution $\widetilde{x}$ of $(\text{CVaR}_\varepsilon^S)$
Generate S equiprobable samples c^s;
$\mathcal{S} \leftarrow \{1, ..., S\}$;
repeat
  compute the optimal solution $\widetilde{x}$ of $(\partial\text{CVaR}_\varepsilon^{\mathcal{S}})$ with objective value $\widetilde{w}$;
  $\widetilde{C} = \{s \in \{1, ..., S\} : c^s \widetilde{x} - \widetilde{z} > 0\}$;
  $\mathcal{S} \leftarrow \mathcal{S} \cup \widetilde{C}$;
until $\text{CVaR}_\varepsilon^S(c^\top \widetilde{x}) - \widetilde{w} \leqslant \delta$;
return $\widetilde{x}$

Aggregation scheme (CA). In [35] the authors developed a computationally fast aggregation technique to solve the CVaR minimization problem in the case of continuous linear programming problems (i.e.: when X is a polyhedron). We show that their approach is valid as well for any feasible set $X \subseteq \mathbb{R}^m$. Let $\mathcal{Q} = \left\{ \mathcal{Q}^1, ..., \mathcal{Q}^Q \right\}$ be a partition of $\{1, ..., S\}$. Summing of all type (3.8c) constraints corresponding to the scenarios of a single bundle Q^q we obtain the following aggregated constraint:

$$\sum_{s \in \mathcal{Q}^q} \eta_s \geqslant \sum_{s \in \mathcal{Q}^q} p_s \left((c^s)^\top x - z \right) \qquad \forall q \in \{1, ..., Q\}$$

Defining $\widetilde{\eta}_q = \sum_{s \in \mathcal{Q}^q} \eta_s$, $\widetilde{p}_q = \sum_{s \in \mathcal{Q}^q} p_s$ and $\widetilde{c}^q = \widetilde{p}_q^{-1} \sum_{s \in \mathcal{Q}^q} p_s c^s$, we can write this last constraint as follows:

$$\widetilde{\eta}_q \geqslant \widetilde{p}_q \left[(\widetilde{c}^q)^\top x - z \right] \qquad \forall q \in \{1, ..., Q\}$$

Given that the $\widetilde{\eta}_s$ are built upon a partition the following problem is a relaxation of (3.8):

$$\widetilde{w}_{\mathcal{Q}} := \min_{z, \widetilde{\eta}, x} z + \varepsilon^{-1} \sum_{q=1}^{Q} \widetilde{\eta}_q \tag{3.15a}$$

$$(\text{CVaR}_\varepsilon^{\mathcal{Q}}) \quad \text{s.t. } : x \in X \tag{3.15b}$$

$$\widetilde{\eta}_q \geqslant \widetilde{p}_q \left((\widetilde{c}^q)^\top x - z \right) \qquad \forall q \in \{1, ..., Q\} \tag{3.15c}$$

$$\widetilde{\eta} \geqslant 0 \tag{3.15d}$$

Consequently we have $\widetilde{w}_{\mathcal{Q}} \leqslant w_S$, which is tight when the optimal solution $\widetilde{x}$ of (3.15) is optimal as well for (3.8). On the contrary, when $\widetilde{w}_{\mathcal{Q}} < w_S$, we can refine the constraints (3.15c) when evaluating the optimal solution of problem (3.15). [35] propose the following refinement scheme: start with $\mathcal{Q} = \{\{1, ..., S\}\}$ and solve ($\text{CVaR}_{\varepsilon}^{\mathcal{Q}}$), then get its optimal solution $\widetilde{x}$'s estimated objective value $\text{CVaR}_{\varepsilon}^{S}(c^{\top}\widetilde{x})$ and its induced partition $\mathcal{Q}_{\widetilde{x}}$. If $\text{CVaR}_{\varepsilon}^{S}(c^{\top}\widetilde{x}) - \widetilde{w}_{\mathcal{Q}} \leqslant 10^{-6}$, we stop because $\widetilde{x}$ is optimal with respect to machine precision. Otherwise we refine the partition $\mathcal{Q} \leftarrow \{\mathcal{Q}_{\widetilde{x}}^{i} \cap \mathcal{Q}^{j} : \mathcal{Q}_{\widetilde{x}}^{i} \in \mathcal{Q}_{\widetilde{x}}, \mathcal{Q}^{j} \in \mathcal{Q}\}$ and iterate. We remark that when refining $\mathcal{Q}$ we always obtain another partition of $\{1, ..., S\}$ with a greater or equal number of subsets. Furthermore, if refining the partition does not change its cardinality we can deduce that for each $\mathcal{Q}_j \in \mathcal{Q}$ we have $\exists i \in \{1, 2, 3\} : \mathcal{Q}^{j} \subset \mathcal{Q}_{\widetilde{x}}^{i}$. This implies that the solution $(\widetilde{x}, \widetilde{z}, \bar{\eta})$ with $\bar{\eta}_s := \widetilde{\eta}_q$, $\forall s \in \mathcal{Q}_q$ is feasible for (3.8) and has the same objective value as the relaxed problem (3.15), implying its optimality.

The algorithm solves at most S aggregated problems until $\mathcal{Q} = \{1, ..., S\}$, where the problem becomes exactly the one in (3.8). Aside from this worst case, we expect from this procedure to solve only a few problems with reduced size instead of a brute force resolution of the complete problem. In [35] they show that for LP problems this procedure can be several orders of magnitude better than standard methods when using LP-tailored approaches. We show a pseudo code of this algorithm in Algorithm 4.

Algorithm 4: Pseudocode of CA algorithm

Data: A problem ($\text{CVaR}_{\varepsilon}^{S}$)
Result: A δ-optimal solution $\widetilde{x}$ of ($\text{CVaR}_{\varepsilon}^{S}$)
Generate S equiprobable samples c^s;
$\mathcal{Q} \leftarrow \{\{1, ..., S\}\}$;
repeat
  compute the optimal solution $\widetilde{x}$ of ($\text{CVaR}^{\mathcal{Q}}$) with objective value $\widetilde{w}$;
  $\mathcal{Q} \leftarrow \{\mathcal{Q}_{x}^{i} \cap \mathcal{Q}^{j} : \mathcal{Q}_{x}^{i} \in \mathcal{Q}_{x}, \mathcal{Q}^{j} \in \mathcal{Q}\}$;
until $\text{CVaR}_{\varepsilon}^{S}(c^{\top}\widetilde{x}) - \widetilde{w} \leqslant \delta$;
return $\widetilde{x}$

Our implementation of these two methods (CA and CS) considers the case when X has integer solutions, given by the shortest path polytope. We tested two different frameworks: one solving directly the MIP, and another one solving first the LP relaxation - obtaining a partition for CA and a set of subgradient cuts for CS - and then solve the MIP with the same algorithm but starting with the constraints added during the first step for CS or the partition returned by the LP for CA. We will denote CA/IP(CS/IP) the direct resolution of the integer programming problem and CA/LP+IP(CS/LP+IP) the latter way.

3.2.2 Entropic Risk Measure

We want to solve the following optimization problem:

$$\min_{x \in X} \mathrm{E}_{\alpha}\left(c^{\top}x\right) := \alpha \ln \mathbb{E}\left[e^{\frac{c^{\top}x}{\alpha}}\right] \tag{3.16}$$

Uncorrelated case In the case where the variables are uncorrelated, the problem (3.16) can be solved with a single run of any shortest path algorithm. Indeed, if random variables are independent, the expected value of their product is the product of their expected values and we have:

$$\mathrm{E}_\alpha\left(c^\top x\right) = \alpha \sum_{i=1}^{m} \ln \mathbb{E}\left[e^{\frac{c_i x_i}{\alpha}}\right]$$

Noticing that:

$$\ln \mathbb{E}\left[e^{\frac{c_i x_i}{\alpha}}\right] = \begin{cases} 0 & \text{If } x_i = 0 \\ \ln \mathbb{E}\left[e^{\frac{c_i}{\alpha}}\right] & \text{If } x_i = 1 \end{cases}$$

we can solve (3.16) by solving the equivalent following problem:

$$\min_{x \in X} \alpha \sum_{i=1}^{m} x_i \ln \mathbb{E}\left[e^{\frac{c_i}{\alpha}}\right]$$

which is a standard shortest path problem with arc costs given by $\alpha \ln \mathbb{E}\left[e^{\frac{c_i}{\alpha}}\right]$. This expected value can be approximated by the following SAA formulation:

$$\min_{x \in X} \left\{ \mathrm{E}_\alpha^S\left(c^\top \widetilde{x}\right) := \alpha \sum_{i=1}^{m} x_i \ln \sum_{s=1}^{S} p_s e^{\frac{c_i^s}{\alpha}} \right\} \tag{3.17}$$

In practice, C programming language's 'double' type accepts numbers up to 10^{308} so the sums $\sum_{s=1}^{S} p_s e^{\frac{c_i^s}{\alpha}}$ cannot be allowed to be greater than 10^{308}. In order to give more slack to the values of α used, i.e.: be able to tackle cases with smaller values of α, for each edge $i \in \{1, ..., m\}$ we compute $\bar{c}_i = \max_{s \in \{1,...,S\}} c_i^s$ which can be used to write the following equivalence:

$$\mathrm{E}_\alpha^S\left(c^\top \widetilde{x}\right) = \alpha \ln \left[\sum_{s=1}^{S} p_s e^{\frac{c_i^s}{\alpha}}\right] = \bar{c}_i + \alpha \ln \left[\sum_{s=1}^{S} p_s e^{\frac{c_i^s - \bar{c}_i}{\alpha}}\right] \tag{3.18}$$

Noticing that $\frac{c_i^s - \bar{c}_i}{\alpha} \leqslant 0$, the exponential values will then always be below one, turning the problems numerically tractable.

We will not consider this formulation further in our work as it heavily relies on the absence of correlation between the random variables. Nevertheless, the scaling method we just described will be extensively used in the next algorithms.

Subgradient (ES). Similar to CVaR, given that the function $\mathrm{E}_\alpha(c^\top \cdot) : x \longmapsto \alpha \ln \mathbb{E}\left[e^{\frac{c^\top x}{\alpha}}\right]$ is convex, so we can represent it as the point-wise maximum of its tangent planes. We can write the subgradient-cuts formulation of the SAA version of (3.16) as follows:

$$\min_{x \in X, t \in \mathbb{R}} \left\{ t : t \geqslant \alpha \ln \left[\sum_{s=1}^{S} p_s e^{\frac{(c^s)^\top x^0}{\alpha}}\right] + \left(\frac{\sum_{s=1}^{S} p_s e^{\frac{(c^s)^\top x^0}{\alpha}} c^s}{\sum_{s=1}^{S} p_s e^{\frac{(c^s)^\top x^0}{\alpha}}}\right)^\top \left(x - x^0\right), \quad \forall x^0 \in X \right\} \tag{3.19}$$

In contrast with CVaR, formulation 3.19 has an infinite number of constraints. We can use the iterative Algorithm 3 to solve the problem with a given tolerance. Indeed, at each iteration k we can add the subgradient cut corresponding to the current incumbent solution x^k and compare the lower bound it returns with the sampled value of the incumbent solution to evaluate the optimality gap. Again, there are scaling issues with the computation of the entropy and its gradient values so the cut we add at each iteration can be equivalently transformed to:

$$t \geqslant m_k + \alpha \ln \left[\sum_{s=1}^{S} p_s e^{\frac{(c^s)^\top x^k - m_k}{\alpha}} \right] + \left(\frac{\sum_{s=1}^{S} p_s e^{\frac{(c^s)^\top x^k - m_k}{\alpha}} c^s}{\sum_{s=1}^{S} p_s e^{\frac{(c^s)^\top x^k - m_k}{\alpha}}} \right)^\top \left(x - x^k \right) \tag{3.20}$$

With $m_k := \max_{s \in \{1,\dots,S\}} \left\{ (c^s)^\top x^k \right\}$. As for CVaR we notice that the first step with only one gradient cut is equivalent to solving a deterministic version of the original problem and we can solve it with Dijkstra's algorithm because of the positiveness of the gradient of the entropic risk measure. As for algorithms CS and CA, we tested the IP and LP+IP frameworks for ES.

Exponential approximation (EEA). Since $\alpha \ln(\cdot)$ is nondecreasing we have that minimizing the entropic risk measure is equivalent to solving the following problem:

$$\min_{x \in X} \sum_{s=1}^{S} p_s e^{\frac{(c^s)^\top x}{\alpha}} \tag{3.21}$$

This last method consists in approximating each of the exponentials rather than the entire function. The classical way to approximate a convex function is to define *a priori* a set $(t_k)_{k \in \{0,\dots,K\}}$ of $K+1$ points on the domain $[l, u]$ of the function to approximate such that $l = t_0 < t_1 < \dots < t_{K-1} < t_K = u$. Adding an extra continuous variable for each scenario, we approximate each exponential by the point-wise maximum of its tangents in the discretization points as follows:

$$\widetilde{w}_K := \quad \min_{z,x} \sum_{s=1}^{S} p_s z_s \tag{3.22a}$$

$$(E_K^S) \quad \text{s.t. } : x \in X \tag{3.22b}$$

$$z_s \geqslant e^{t_k} \left[1 + \frac{(c^s)^\top x}{\alpha} - t_k \right] \qquad \forall s \in \{1, \dots, S\}, \forall k \in \{1, \dots, K\} \tag{3.22c}$$

Which is a relaxation of the original problem since tangents are always under the function for convex functions. One can choose the discretization such that the error of approximation is arbitrarily small at the cost of the number of breakpoints. Nevertheless, there are several reasons to limit the number of breakpoints for our discretization: 1) each scenario must have its own discretization, adding together $S(K+1)$ extra constraints and S extra continuous variables, 2) the original path polyhedron can already be large, 3) the interval $[l, u]$ can be

particularly large if the random variables have a high variability and 4) scaling issues can be prohibitive: as u grows e^{t_k} can be out of the acceptable range of modern computing abilities.

Our approach consists in premultiplying the whole objective function by $e^{-\frac{M}{\alpha}}$ with $(c^s)^\top x \leqslant M, \forall x \in X, \forall s \in \{1, ..., S\}$ and use an approximation of $t \longmapsto e^t$ on $]-\infty, 0]$ resulting in the following formulation:

$$\widetilde{w}_K(M) := \min_{z,x} \sum_{s=1}^{S} p_s z_s \tag{3.23a}$$

$$(E_K^S(M)) \quad \text{s.t. } : x \in X \tag{3.23b}$$

$$z_s \geqslant e^{t_k}\left[1 + \frac{(c^s)^\top x - M}{\alpha} - t_k\right] \qquad \forall s \in \{1, ..., S\} \tag{3.23c}$$

$$\forall k \in \{1, ..., K\}$$

We notice that the exponential can be well approximated over $]-\infty, 0]$ with a fairly small number of points. Using the points $t_k := 2\ln\left(\frac{k}{K}\right)$ with $K = 10$ gives an approximation of maximum absolute error of $6.15 \cdot 10^{-3}$. Due to the structure of the exponential function, we have that the optimal solution of formulation (3.23) gives a lower bound for problem (3.16). Indeed, we have that:

$$e^{\frac{M}{\alpha}}\widetilde{w}_K(M) = \min_x \sum_{s=1}^{S} p_s \max_{k\in\{1,...,K\}} \left\{e^{t_k + \frac{M}{\alpha}}\left[1 + \frac{(c^s)^\top x}{\alpha} - \left(t_k + \frac{M}{\alpha}\right)\right]\right\} \tag{3.24}$$

Which is a piecewise linear lower approximation of the exponential function with points $\left(t_k + \frac{M}{\alpha}\right)_{k\in\{1,...,K\}}$. Further, given that $\sum_{s=1}^{S} p_s = 1$ the absolute error of the approximated objective function is the same as the individual absolute error across the scenarios. Figure 3.4 illustrates the approximations done here.

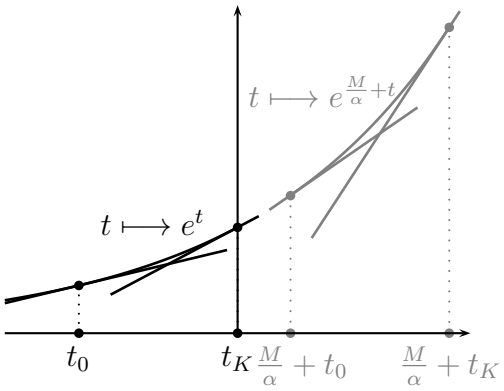


Figure 3.4: Lower piecewise linear approximations of $t \longmapsto e^t$ and $t \longmapsto e^{\frac{M}{\alpha}+t}$

Given that $e^{-M} > 0$, the resulting problem to solve is equivalent to (3.22). Still, although this approach has no scaling issues it loses numerical precision in practice because of big values of M. We can see that taking $M^* := \max_{s\in\{1,...,S\}} (c^s)^\top x^*$ with x^* the optimal solution of the real problem (3.16), then solving (3.23) must return x^* as every $\frac{(c^s)^\top x - M^*}{\alpha}$ is negative,

i.e.: where the exponential is correctly approximated. This suggests an iterative search of the value of M^*, successively solving (3.23) and updating $M := \max_{s\in\{1,...,S\}} (c^s)^\top \widetilde{x}$ with $\widetilde{x}$ the last optimal solution. We repeat the process until the gap between the estimated entropic value of $\widetilde{x}$, $\mathrm{E}_\alpha^S\left(c^\top \widetilde{x}\right)$ and $M + \alpha \ln \widetilde{w}_K(M)$ is smaller than a given tolerance δ. In practice, there is a last computational issue with the lower bound returned by the approximated problem (3.23). Actually, CPLEX solver considers zero every value smaller than 10^{-6} so in reality the solver stops whenever the reduced costs are smaller than this tolerance. Hence, there are two issues with the solution returned: 1) The absolute gap of the solution returned by the relaxed problem is a $10^{-6}e^{\frac{M}{\alpha}}$ approximation of the linear approximation problem and hence, can be suboptimal and can fail to provide a lower bound. and 2) The approximation provides a solution of objective value zero hence providing a $-\infty$ lower bound. This can be tackled considering that the value $e^{\frac{M}{\alpha}}\widetilde{w}_K(M)$ is a genuine lower bound whenever $\widetilde{w}_K(M) > 10^{-6}$. Because of these computational problems, there is no guarantee that EEA returns an optimal solution for (E_α^S). We show a pseudo code in Algorithm 5.

Algorithm 5: Pseudocode of EEA algorithm

Data: A problem (E_α^S)
Result: A feasible solution $\widetilde{x}$ of (E_α^S)
Generate S equiprobable samples c^s;
$\widetilde{x} \in \arg\min\left\{\sum_{s=1}^{S} p_s \left(c^s\right)^\top x : x \in X\right\}$;
$M \leftarrow \max\left\{\left(c^s\right)^\top \widetilde{x} : s \in \{1,...,S\}\right\}$;
repeat
  Solve $(\mathrm{E}_K^S(M))$ of 'optimal' solution $\widetilde{x}$ and objective value $\widetilde{w}$;
  $M \leftarrow \max_{s\in\{1,...,S\}} (c^s)^\top \widetilde{x}$;
until $(\widetilde{w} > 10^{-6}) \wedge \left(E_\alpha^S\left(c^\top \widetilde{x}\right) - \widetilde{w} \leqslant \delta\right)$;
return $\widetilde{x}$ is a feasible solution of (E_α^S).

3.3 General solution framework

Now that we can formulate the sample average approximation of our problems, we want to guarantee how close the solution to this approximate problem is to the real solution. In this goal, [44] developed a framework successively solving SAA problems for discrete stochastic optimization problems giving computational bounds on the real (not sampled) optimization problem (3.2). In what follows we assume that the scenarios are uniformly distributed, i.e.: $p_s = \frac{1}{S}$ for any scenario $s \in S$.

3.3.1 Stochastic lower bound

The optimal sampled value w_S is a random variable whose realizations depend on the samples that were selected. Its relation to the optimal value of (3.2), w^*, is given by (3.25) due to

the convexity of the expected value:

$$\mathbb{E}\left[w_S\right] \leqslant w^* \tag{3.25}$$

We can estimate this expected value averaging the optimal values of problem P_S generating several samples of size S and solving the associated problem. Let T be the number of times we generate a different set of samples and solve P_S. Then, given a confidence level $\gamma \in [0,1]$ and the objective values $(w_S^t)_{t\in\{1,\dots,T\}}$ of the T different problems, by the central limit theorem we have:

$$\mathbb{P}\left[L_{N,T} := \frac{1}{T}\sum_{t=1}^{T} w_S^t - \phi^{-1}\left(1-\frac{\gamma}{2}\right)\frac{1}{\sqrt{T}}\sigma_S \leqslant w^*\right] \geqslant 1-\frac{\gamma}{2} \tag{3.26}$$

With:

$$\sigma_S^2 = \frac{1}{T-1}\sum_{t=1}^{T}\left(w_S^t - \frac{1}{T}\sum_{t'=1}^{T} w_S^{t'}\right)^2 \tag{3.27}$$

Namely, the resolution of several sampled problems gives a lower bound of the real optimal value with confidence at least $1-\frac{\gamma}{2}$. When using suboptimal resolutions with known relative gaps $(g_t)_{t\in\{1,\dots,T\}}$ and objective values $(w_S^t)_{t\in\{1,\dots,T\}}$, notice that we cannot directly multiply each sampled objective value w_S^t by $\frac{1}{1+g_t}$ because of the correcting term containing the standard deviation. One way to tackle this issue is to adjust the sampled objective values by a factor of $\frac{1}{1+\max\{g_t : t\in\{1,\dots,T\}\}}$.

3.3.2 Stochastic upper bound

When solving the T sampled problems we can keep some feasible solution $x \in X$ and we know that $w^* \leqslant \mathbb{E}\left[u\left(c^\top x\right)\right]$. Now, we can have an estimation of $\mathbb{E}\left[u\left(c^\top x\right)\right]$ generating another sample of size $S' >> S$. It is common practice to refer to the sample S used for each of the sampled problems solved for the lower bound as the "in-sample" and this larger sample S' as the "out-of-sample". Again, by the central limit theorem we have that:

$$\mathbb{P}\left[w^* \leqslant U_{S'}(x) := w_{S'}(x) + \phi^{-1}\left(1-\frac{\gamma}{2}\right)\frac{1}{\sqrt{S'}}\sigma_{S'}(x)\right] \geqslant 1-\frac{\gamma}{2} \tag{3.28}$$

With:

$$\sigma_{S'}^2(x) := \frac{1}{S'-1}\sum_{s=1}^{S'}\left[u\left(\left(c^s\right)^\top x\right) - w_{S'}(x)\right]^2 \tag{3.29}$$

In other words, it is possible to obtain an upper bound of the real optimal value by computing an accurate estimation of the objective value of some feasible solution. Now that we have lower and upper bounds we can compute the stochastic gap, i.e.: the real gap of our solution $x \in X$ with the real optimal value with confidence at least $1-\gamma$:

$$\mathbb{P}\left[L_{N,T} \leqslant w^* \leqslant U_{S'}(x)\right] \geqslant 1-\gamma \tag{3.30}$$

A common practice is to store the solution corresponding to the lowest objective value w_S^t returned by one of the T deterministic equivalent problems and use it to build the stochastic upper bound.

3.4 Computational Experiments

3.4.1 Experimental Set-up

Network generation. We generated grid networks of $r \times r$ nodes with $r \in \{5, 7, 10, 13, 15\}$. Each network has exactly the same geographical dimensions (a square with sides of 1.5 kilometer on each side), the only difference residing in the topology of the network. In addition to the grid roads, each instance includes a 'highway' that can be of three types as depicted in figure 3.5: straight crosses (dashed), tilted crosses (dotted) or ring-shaped (solid).

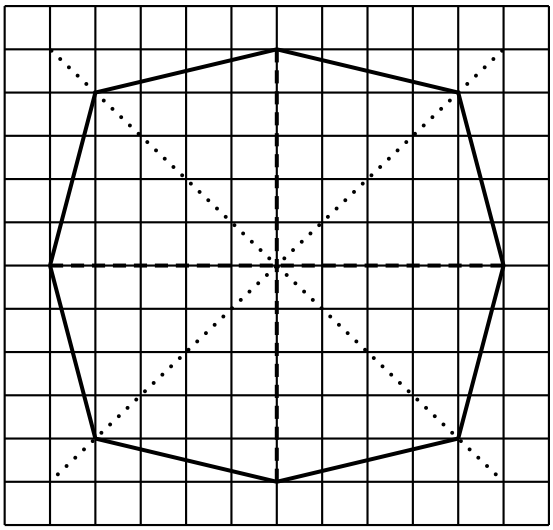


Figure 3.5: Example of Grid networks generated for $r = 13$

Distributions generation. For each edge $i \in E$ we considered a constant mean speed μ_i of 50 km (80 km) for grid edges (highway edges). To generate more instances we perturbed these reference values with an uniform noise $u_i \sim \mathcal{U}[0.5, 1.5]$ such that the resulting mean speed on edge i is $\mu'_i := \mu_i u_i$. For each instance, we generated ten times a perturbation with a different random seed. We also generated randomly special covariance matrices $\Sigma = (\sigma_{ij})_{(i,j)\in\{1,\dots,m\}^2}$, where $\sigma_{ii} = \sigma_i^2$ is the variance of the random variable associated with edge $i \in \{1, ..., m\}$. Given a vector of expected values $\mu' = (\mu'_i)_{i\in\{1,\dots,m\}}$ we considered instances with different coefficients of variation $c \in \{0.5, 2\}$ for normal streets and $c \in \{1, 4\}$ for highways (riskier) such that the standard deviations are defined as $\sigma_i = c\mu'_i$. We considered the uncorrelated and correlated cases. We built the correlations in a way that all the highway edges (street edges) are positively correlated with each other, but there is negative correlation between highway and street edges. This will highlight the risk-hedging effect obtained by forcefully passing through both types of edges in spite that one type is riskier than the other. We generated correlation matrices C defined by a product $C = LL^\top$ with $L = (l_{ij})_{(i,j)\in\{1,\dots,m\}^2}$ a lower triangular matrix whose rows are unitary vectors. Given a covariance matrix C and

standard deviations $(\sigma_i)_{i\in\{1,\dots,m\}}$ we can then compute a covariance matrix Σ as follows:

$$\Sigma = \text{Diag}\left((\sigma_i)_{i\in\{1,\dots,m\}}\right) \cdot C \cdot \text{Diag}\left((\sigma_i)_{i\in\{1,\dots,m\}}\right) \tag{3.31}$$

Which has all the required properties to be a correlation matrix. We considered that travel times on graph edges are random following Log-normal distributions as indicated by [82] in the case of transportation networks, of joint-parameters (p, S) with $p = (p_i)_{i\in\{1,\dots,m\}}$ and $S = (s_{ij})_{(i,j)\in\{1,\dots,m\}^2}$. Given some expected value vector μ and a covariance matrix Σ we wanted to generate samples from the associated log-normal distribution. We know that such a distribution is a log-normal of parameters (p, S) with:

$$\begin{cases} p_i &= \ln\left(\frac{\mu_i^2}{\sqrt{\mu_i^2+\sigma_{ii}}}\right) \\ s_{ij} &= \ln\left(1+\frac{\sigma_{ij}}{\mu_i\mu_j}\right) \end{cases} \tag{3.32}$$

Moreover, given some decomposition $S = L^\top L$ we have $LX+p \sim \mathcal{N}(p, S)$ with $X_i \sim \mathcal{N}(0, 1)$. Putting everything together, we can sample multivariate log-normal distributions of mean vector μ and covariance matrix Σ by generating univariate standard normal samples $x = (x_i)_i$ and applying the transformation $z = \exp(Lx + p)$. We generated the standard normal distribution samples via Box-Muller transform: given u and u' two independent draws of an uniform law $\mathcal{U}[0, 1]$ then:

$$x = \sqrt{-2\ln u}\cos(2\pi u')$$
$$x' = \sqrt{-2\ln u}\sin(2\pi u')$$

Are distributed as independent $\mathcal{N}(0, 1)$ random variables.

Parameters used. We chose the sample sizes used for each deterministic equivalent problem (in-sample) to be in $S \in \{0.5, 1, 2, 5, 10\} \cdot 10^3$ and the sample used for upper bounding (out of sample) in $S' \in \left\{(2k)_{k\in\{1,\dots,10\}}\right\} \cdot 10^4$. The base problem we want to solve is to find the less risky path between the north-western and the south-eastern corners of each grid network. We tested CVaR_ε for $\varepsilon \in \{1, 5, 10, 50, 90, 95, 99\}\%$. For the entropic risk measure, the parameter α has units (like travel time) so we had to carefully choose its values. We chose a reference path inside the grid network of 15×15 nodes with circular highway and computed the sampled cumulative distribution of the travel time on that path, $F(\cdot)$, with correlated travel times across edges and a high dispersion factor ($c = 2$). We then chose $\alpha \in F^{-1}(\{0.01, 0.05, 0.1, 0.5, 0.9, 0.95, 0.99\})$, whose rounded computed values are: $\alpha \in \{10, 15, 20, 50, 200, 250, 400\}$ $[s]$. The parameter α captures an absolute risk aversion and penalizes greatly paths whose random travel time exceeds α. This is why we kept the same geographical dimensions for the grid networks to keep the same set of parameters for testing all the instances.

Experiments. To evaluate the influence of each parameter on the efficiency of the proposed algorithms, we defined a base case and conducted experiments that let varied one parameter at a time. First, we wanted to compare the computational efficiency of each solution algorithm

on deterministic equivalent problems. We took as a base case the grid network of 10×10 nodes with circular highway, presence of correlation, high dispersion and a sample size of $S = 2000$. If an algorithm does not reach the optimal solution within 250 minutes, we return the incumbent solution with its respective optimality gap. We then chose the algorithm with best computational behavior for each risk measure and used it to solve $T = 50$ experiments with samples of size $S = 2000$ with a time limit of 250 minutes for the whole run and 50 minutes for each deterministic equivalent problem solution. We computed the stochastic bounds with 95% confidence (i.e. with $\gamma = 0.05$).

Risk measure	Algorithm	Framework	Abbreviation
CVaR	Monolithical resolution	-	CM
	Subgradient Algorithm	LP+IP	CS/LP+IP
		IP	CS/IP
	Aggregation method	LP+IP	CA/LP+IP
		IP	CA/IP
Entropy	Subgradient algorithm	LP+IP	ES/LP+IP
		IP	ES/IP
	Exponential approximation method	-	EEA

Table 3.1: Frameworks abbreviations

3.4.2 Computational results

Deterministic Equivalent problems. In this section, we will present computational results comparing the efficiency of the different methods we presented earlier. In this goal, we compared the runtimes and the optimality gaps of the solutions returned by each of the algorithms over a single deterministic equivalent problem. This determined which method is most fit to compute stochastic bounds later. In table 3.2 we show averaged results over ten random instances for each cell. We see that the monolithic version of CVaR has largest solution times than the other algorithms, except when there are no correlations. We see that CA/IP outperforms greatly all the other algorithms in terms of execution time. In the general correlated case, we can rank - on average - the CVaR algorithms proposed here as follows: [CA/IP] $\succeq$ [CA/LP+IP] $\succeq$ [CS/IP] $\succeq$ [CS/LP+IP] $\succeq$ [CM]. For the entropic risk measure, we can see that ES/IP is always much faster than the other algorithms. EEA behaves slowly in practice due to the number of iterations needed to find an adequate M^* and then approximate correctly the exponentials. Furthermore, the optimality gaps are often strictly positive because there is no guarantee that the stopping criterion ensures that EEA returns an optimal solution. On average, we can rank the Entropy frameworks as follows: [ES/IP] $\succeq$ [ES/LP+IP] $\succeq$ [EEA]. In the end, first solving the LP relaxation and then the original problem with the constraints generated is always slower. Although the number of iterations is smaller for LP+IP frameworks, the fact they begin the integer resolution with a larger problem instead of starting with a small one slows down significantly the algorithm. The correlations have a great influence over the execution time. Indeed, we designed Σ such that alternating between both types of streets helps to hedge against variability and thus reduces the optimal region, speeding up the procedure. The correlation also groups several edges

Param	Value	CM	CA/LP+IP	CA/IP	CS/LP+IP	CS/IP	EEA	ES/LP+IP	ES/IP
Σ	I	**59.1**	195.6	154.7	1392(8)	1313(10.9)	1658.7(1.8)	1318.7(5.6)	**1312.5(4.4)**
	$\neq I$	15.3	2.2	**1.9**	4.2	4.1	1243.7(3.6)	4.35	**3.1**
c	0.5	10.9	1.8	**1.5**	2	2	849.1(2.3)	2.1	**1.7**
	2	15.3	2.2	**1.9**	4.2	4.1	1135.3(3.9)	4.4	**3.1**
Shape	+	13.6	2.2	**1.9**	6	5	1173.8(2.9)	7	**4**
	×	15.2	2.2	**2**	6.9	6.6	1287.5(3.3)	9.3	**5.7**
	O	15.3	2.2	**1.9**	4.2	4.1	1135.3(3.9)	4.4	**3.1**
r	5	3.6	0.3	**0.1**	0.5	0.3	104.9(6.4)	0.3	**0.2**
	7	7.5	0.7	**0.4**	1.1	0.9	329.6(3.8)	1	**0.6**
	10	15.3	2.2	**1.9**	4.2	4.1	1135.3(3.9)	4.4	**3.1**
	13	24.8	4.5	**4.4**	6.9	6.3	2043.1(5.2)	9.9	**8.7**
	15	33.9	**7.7**	**7.7**	10.4	9.9	1890.8(6.3)	17	**13.2**
S	500	2.1	**0.6**	0.7	2.4	2.8	156.9(1.9)	4.3	**2.7**
	1000	6	1.3	**1.2**	3.3	3.7	379(3.6)	2.5	**1.9**
	2000	15.3	2.2	**1.9**	4.2	4.1	1135.32(3.9)	4.4	**3.1**
	5000	58.3	5	**4.2**	6.6	6.3	cutoff(2)	12.8	**8.3**
	10000	177.5	9.5	**7.7**	10.7	9.6	cutoff(7.3)	19.6	**10.5**

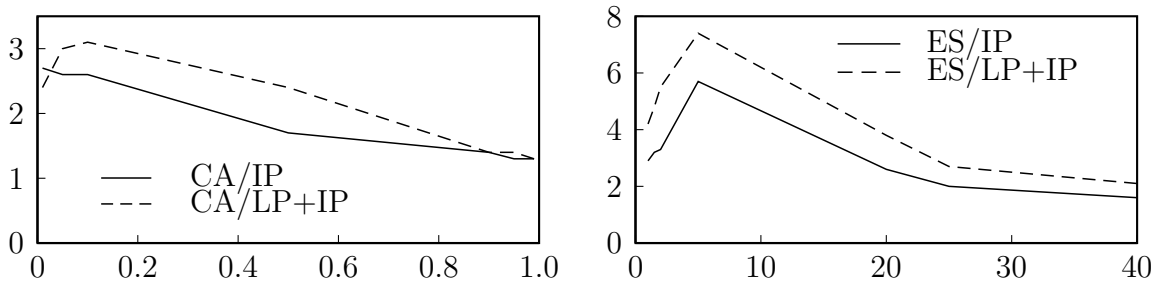
Table 3.2: Average solution time for different instance parameters (time[s](gap[%]))

together in their behaviors, having the solver aggregating variables and making branching more efficiently. In a future work, we shall investigate the influence of weaker correlations. As expected, having more dispersion makes the problem harder as we are getting further from an easy expected value minimization problem. The instance parameters that are the most influential to the solution times are the network size and the number of samples we used for the SAA. The network size being directly linked to the number of integer variables, the execution time grows significantly fast with r. In a similar way, each sample used adds an extra constraint and an extra continuous variable to CVaR problems. The execution time for CA/IP grows linearly with S. For the entropy, although the number of samples does not enlarge the problem solved by ES, it does substantially increase the solution time. In

Param	Value	CM	CA/LP+IP	CA/IP	CS/LP+IP	CS/IP	Param	Value	EEA	ES/LP+IP	ES/IP
ε	0.01	16.6	**2.5**	2.6	9.7	11.4	α	1000	1733.5	4.2	**2.9**
	0.05	19.9	3.0	**2.6**	7.0	6.5		1500	1511.5	4.8	**3.2**
	0.1	20.4	3.1	**2.6**	5.8	4.4		2000	1435	5.5	**3.3**
	0.5	19.3	2.4	**1.7**	2.3	1.9		5000	1117.2(1)	7.4	**5.7**
	0.9	11.0	1.4	**1.4**	1.5	1.5		20000	802.1(12.2)	3.8	**2.6**
	0.95	10.2	1.4	**1.3**	1.4	1.4		25000	728.4(8.3)	2.7	**2.0**
	0.99	9.9	1.3	**1.3**	1.4	1.4		40000	619.6(5.4)	2.1	**1.6**

Table 3.3: Algorithms comparison Vs. risk measure parameters (time[s](gap[%]))

table 3.3, we can see once more than CA/IP (ES/IP) is the best scheme for CVaR (Entropy) resolution as we vary the risk measure parameters. In the following we used these two algorithms to solve the repeated procedure with stochastic bounds computation. When loosening the risk aversion, we observe that the difficulty lowers significantly for the less conservative parameters (see figure 3.6). Indeed, when the parameter changes to reduce risk aversion the problem to solve tends to a determininistic problem which - in our particular case - has a particularly nice structure. Nevertheless, we see that for both risk measures(see figures 3.6a and 3.6b) the most difficult problems to solve are not necessarily the most risk averse. We remark that entropy minimization problems are slightly slower to solve than CVaR.



(a) CA time $[s]$ Vs. $\varepsilon \in]0, 1]$ **(b)** ES time $[s]$ Vs. $\alpha > 0$

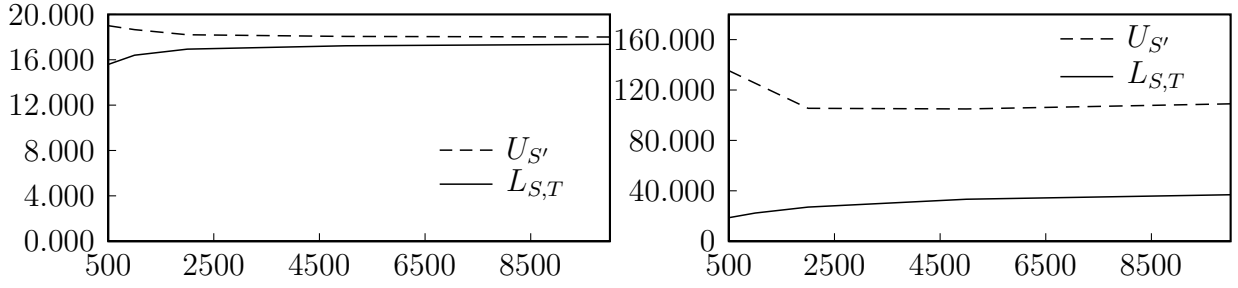
Figure 3.6: Execution time $[s]$ Vs. parameter and LP+IP or IP frameworks.

Stochastic bounds. In this section, we will present computational results comparing the quality of the solutions returned by the entire SAA framework. We ran the T experiments

Param	Value	gap[%] co	gap[%] unco	Param	Value	gap[%] co	gap[%] unco
	0.01	11.87	46.24		1000	68.26	81.04
	0.05	4.91	38.78		1500	69.23	87
	0.1	4.12	34.53		2000	71.05	86.31
ε	0.5	2.71	29	α	20000	90.58	96
	0.9	1.8	19.28		25000	50.31	93.38
	0.95	1.77	19.1		5000	56.92	94.56
	0.99	1.75	18.93		40000	13.74	67.66

Table 3.4: Price of correlation for one instance of the base case.

for both measures and confirmed that solving the problems without considering correlations can lead to solutions of poor quality. In table 3.4 we can see that for a random instance of the base case, an uncorrelated solution can have a difference of optimality gap between [17;43]%([5;53]%) when minimizing CVaR(Entropy). In figure 3.7, we can see that the sample size S is important for two reasons: it approximates better the original problem, improving the lower bound and returns better candidate solutions, improving the upper bound. In

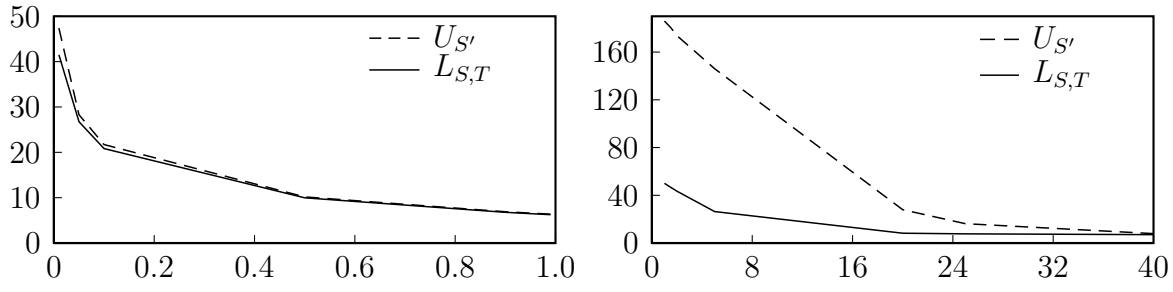


(a) Bounds for CVaR Vs. S **(b)** Bounds for Entropy Vs. S

Figure 3.7: Stochastic optimality bounds Vs. number of samples S.

figure 3.8, we observe that the risk aversion parameter plays a critical role in the difficulty of our problem. In particular for the entropy, the gap is high for almost all the values of α we considered, closing slowly when the risk aversion lowers. For CVaR, the optimality gap becomes acceptable ($< 5\%$) starting from $\varepsilon \geqslant 0.1$.

Increasing the sample size S' improves (reduces) the quality of the upper bound of CVaR



(a) Bounds For CVaR Vs. $\varepsilon \in]0, 1]$ **(b)** Bounds for Entropy Vs. α

Figure 3.8: Stochastic optimality bounds Vs. Risk Aversion parameter

and is computationally cheap. Nevertheless, for the Entropy the size S' of the out of sample needed to have accurate upper bounds can be potentially large. For conservative parameters α, we observe that the upper bound is still unstable for the values of S' we considered. In figure 3.9, the curve represents the upper bound in function of S' and the dots the contribution of each scenario.

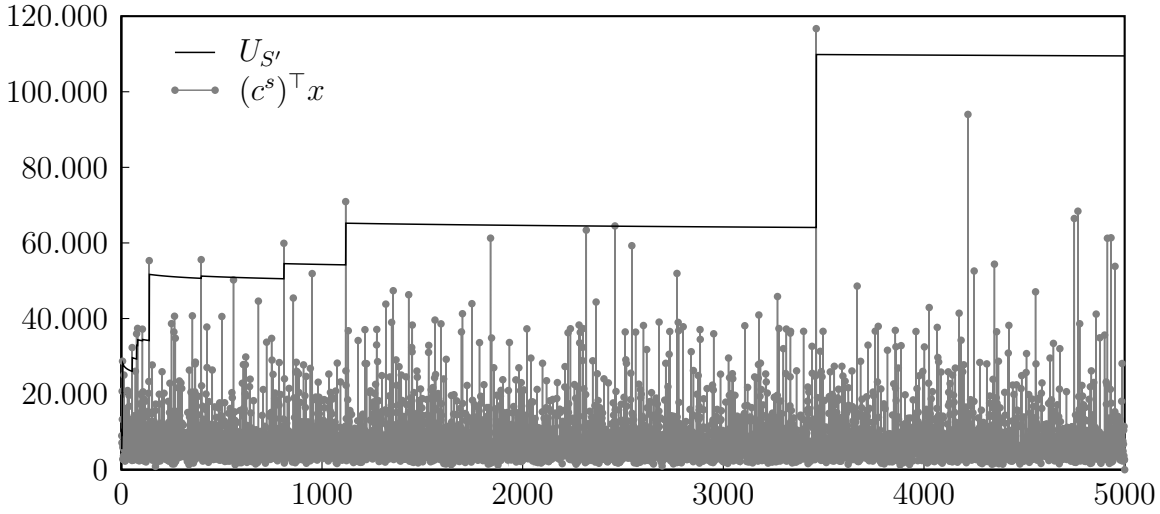


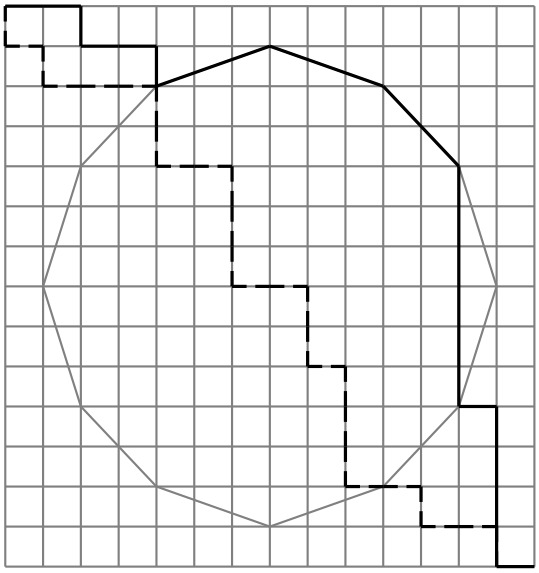
Figure 3.9: Example of upper bound for entropy and scenario realizations $(c^s)^\top x$ Vs. S'

We can see jumps in the upper bound curve between - expected - lowering periods. These jumps appear because the entropic risk measure can be highly sensitive to worst cases. We can show that we have:

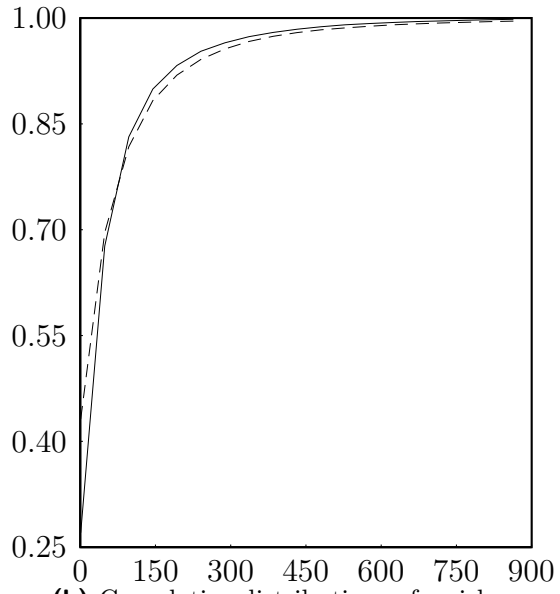
$$\max_{s\in\{1,\dots,S'\}} (c^s)^\top x - \alpha \ln S' \leqslant \alpha \ln \sum_{s=1}^{S'} \frac{1}{S'} e^{\frac{(c^s)^\top x}{\alpha}} \leqslant \max_{s\in\{1,\dots,S'\}} (c^s)^\top x$$

These bounds show that the entropic risk measure highly sensitive to big realizations $(c^s)^\top x$. Instead of having a smoothing effect of big values of S' we can observe the upper bound $U_{S'}$ worsening. We ran a couple of small instances with very large out of samples and could observe that for $S' \approx 10^6$ an expected nonincreasing smooth curve appears.

In figure 3.10a we see an example where the risk averse response (solid) of $\text{CVaR}_{1\%}$ passes trough "highway" edges to hedge against risk, using the negative correlations to its advantage, while the risk neutral (dashed) - minimizing the expected value - passes right where the



(a) Risk averse path Vs. Risk neutral path



(b) Cumulative distributions of a risk averse path Vs. risk neutral path.

Figure 3.10: Comparison of risk averse optimal path for $\text{CVaR}_{1\%}$ and a risk neutral optimal path

mean value is minimal, without taking the volatility of travel times into account. As such, in figure 3.10b we can see their respective cumulative distributions obtained with a large out of sample of size $S' = 100000$. Without surprise, we can notice that the risk averse path (solid) stochastically dominates the risk neutral one (dashed) in the worst 20% of the cases (for both solutions) lowering in a 2% the probability to have a travel time greater than two minutes. Moreover, the expected travel time of a risk averse traveler is about 1 minute 41 seconds against 1 minute 38 for the risk neutral traveler, but the variances are respectively 1 minute 46 seconds for the risk averse traveler and 2 minutes 17 seconds for the risk neutral one.

3.5 Conclusions

In this chapter, we developed a complete framework to solve shortest path problems with uncertainty in the travel times. We developed efficient techniques to solve SAA equivalent problems for the CVaR and the Entropic risk measures. We showed that our algorithms for CVaR minimization return provably decent solutions within a reasonable CPU time. A natural way to tackle the difficulty inherent to big samples and extend these results to real sized networks could come with the use of importance sampling methods, where less samples but more representative are used.

Chapter 4

Reformulations for hard network problems

4.1 Introduction

Including uncertainty into shortest path problems destroys the structure of the original combinatorial problem. With some exceptions, such as the robust SP with interval data ([9]), the SP problem with uncertainty turns NP-hard in most cases ([46]). Consequently, we investigated methods reducing the computational effort required by the solution process. Such a problem can be roughly cast as an integer programming problem finding the path in a network $G = (V, E)$ ($n = |V|$ and $m = |E|$) that minimizes a convex risk measure ρ. In other words, we want to solve the problem (4.1):

$$\min_{x \in X \cap \{0,1\}^m} \rho(x) \tag{4.1}$$

where $X \subseteq [0,1]^m$ is the path polyhedron lattice (4.2):

$$x \in X \;=\; \left\{ x \in [0,1]^m \;:\; \sum_{k:(j,k)\in E} x_{jk} - \sum_{i:(i,j)\in E} x_{ij} = b_j \quad \forall j \in V \right\}, \tag{4.2}$$

where the right hand side vector b satisfies $b_s = 1$, $b_t = -1$, and $b_j = 0$ for all $j \in V \setminus \{s, t\}$. We will assume that in our case the function ρ is such that an optimal solution is cycle free. This occurs for example when $\rho(x) = c^\top x$ with $c \geqslant 0$. This important assumption implies that at most one outgoing (or incoming) edge of each node can be nonzero, in other words we have $\sum_{e \in \delta^+(i)} x_e \leqslant 1$.

Type 1 Special Ordered Set constraints (SOS1, [13]) are a special type of constraints where at most one variable from a subset I can be nonzero (4.3). In the case of binary variables, a

SOS1 is equivalent to:

$$\sum_{i \in I} x_i \leqslant 1 \tag{4.3}$$

$$x_i \in \{0,1\}, \forall i \in I \tag{4.4}$$

This type of constraint has been thoroughly studied in the past decades in the context of disjunctive programming ([12, 11]). Instead of branching over one variable at some Branch and Bound (Branch and Bound) node, one can look for such implicit inequalities in order to branch over a set of variables that can separate more evenly the feasible set ([4, 29, 28, 42]). This idea of constraint branching can be traced back to the late 70' with the work of [36, 68] and is extensively used in column generation algorithms ([75, 77, 76, 59]). A number of network problems admit implicit disjunctive constraints, e.g.: in flow conservation constraints of paths as mentioned before. We extensively used the work of [79] where they present a way to represent generic disjunctive constraints with a logarithmic number of extra binary variables and constraints, generalizing the work in [51, 52]. They prove that for a broad class of hard combinatorial problems this type of representation is as strong as the original one, i.e.: provides equally good Linear Programming (LP) bounds. Further, they point out that it actually simulates a smart constraint branching on the original variables.

We show that it is always possible to enforce the integrality of the original variables with a smaller number of integer variables. In particular, we identify implicit disjunctive constraints allowing to cast the simple path and Hamiltonian circuit sets into formulations having at most $n\left(1+\log_2\left(\frac{m}{n}+1\right)\right)$ integer variables instead of the m integer variables of the original formulation. For the TSP, we extend a formulation of [74] where the disjunctive constraints are explicit.and cast it with a smaller number of zero-one variables of order $O\left(n\log_2\left(\frac{m}{n}\right)\right)$. For *st*-path or *st*-cut problems in Directed Acyclic Graphs (DAGs), we present several ways to generate coverings of all the binary variables by SOS1 constraints, and we present the associated extended formulations for these problems. In particular, we showed that any *st*-path problem in DAGs can be modeled with $O\left(d\log_2\left(\frac{m}{d}\right)\right)$ binary variables, with d the length of the longest path. Similarly, we showed that any *st*-cut problem in DAGs can be modeled with $O\left(f\log_2\left(\frac{m}{f}\right)\right)$ binary variables, with f the size of the maximum cut.

As we mentioned above, the proposed formulations are as strong as the standard formulations, but while they consider less integer variables they have overall more variables and constraints. Therefore the practical benefits of these formulations for computation are not clear. Nevertheless, the proposed formulations can help improve solution methods. Instead of directly solving the proposed formulations, we propose to solve the original models without the additional variables but using a modified branching procedure when calling CPLEX ([27]), explicitly using those implicit inequalities and branch on them.

The rest of the chapter is structured as follows: In the next section we present the fundamental result of [79] that we will extensively use in the whole chapter. Section 4.3 introduces a way to reduce the number of integer variables of any simple path problem, with a direct application to the Travelling Salesman Problem (TSP). We then present several ways to reformulate path or cut problems in DAGs in 4.4 and we finally conclude in section 4.5.

4.2 SOS1 with a logarithmic number of $\{0,1\}$ variables

4.2.1 Equivalent formulation

A result of [78] shows that any disjunctive constraint of the type

$$\sum_{i\in I} x_i = 1 \tag{4.5}$$

$$x_i \in \{0,1\}, \forall i \in I \tag{4.6}$$

Can be expressed equivalently relaxing the integrality in (4.6) and adding $\lceil \log_2 |I| \rceil$ extra zero-one variables and extra constraints.

Proposition 1. *[79]*

$$\begin{cases} \sum_{i\in I} x_i = 1 \\ x_i \in \{0,1\}, \forall i \in I \end{cases} \Leftrightarrow \begin{cases} \sum_{i\in I} x_i = 1 \\ x \geqslant 0 \\ \exists! z \in \{0,1\}^{L(|I|)} : \sum_{p\in S^+_{|I|}(l)} x_p = z_l \quad \forall l \in \{1,...,L\left(|I|\right)\} \end{cases} \tag{4.7}$$

With $L(d) := \lceil \log_2 d \rceil$ *for any* $d \in \mathbb{N}$. *And for any* $l \in \{1,...,L(d)\}$*:*

$$S^+_d(l) := \bigcup_{s=1}^{2^{L(d)-l}} \left(\bigcup_{t=1}^{2^{l-1}} \left(t + 2^{l-1} + (s-1)2^l \right) \right) \tag{4.8}$$

$$S^-_d(l) := I \setminus S^+_d(l) = \bigcup_{s=1}^{2^{L(d)-l}} \left(\bigcup_{t=1}^{2^{l-1}} \left(t + (s-1)2^l \right) \right) \tag{4.9}$$

First, the sets $S^-_d(l)$ and $S^+_d(l)$ are a partition of $\{1,...,2^{L(d)} \geqslant d\}$ for any $l \in \{1,...,L(d)\}$. We can see in figure 4.1 an example of them for $d = 5$:

The intuition behind the sets is that they can define the binary decomposition of any number between one and d. For example, let consider the tautological case where we want to find the index of the third element - which is obviously three - in the array of the last example by binary search. We first try to know if 3 is in the lower or the upper half of the array, finding out that it is in the former implying that $3 \in S^-_d(1)$. Second, we want to find out in which quarter of the array 3 is, hence finding out that $3 \in S^+_d(2)$. The last step indicates that $3 \in S^-_d(3)$, concluding that index 3's index is 3(!).

With this simple observation, when looking for some index we can define a binary variable $z_l \in \{0,1\}$ for each $l \in \{1,...,L(|I|)\}$ representing if the index belongs to $S^+_{|I|}(l)$ or not (i.e.: belongs to $S^-_{|I|}(l)$). It is easy to see that the z variables are in fact the binary decomposition of the index we are looking for. In our case, where our variables x_i are positive and sum one, the only possibility left is to have exactly one of them at one and the rest to zero.

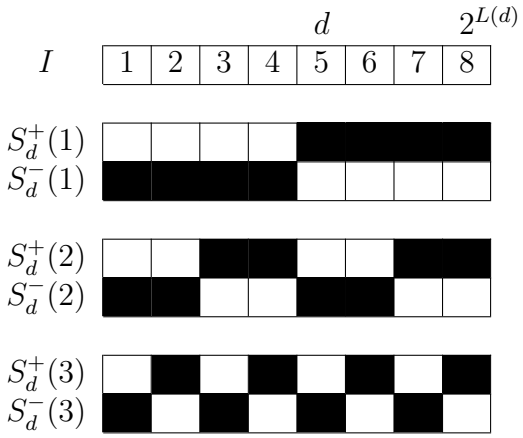


Figure 4.1: Example of partitions $\left(\{S_d^+(l), S_d^-(l)\}\right)_{l\in\{1,..,\lceil\log_2 d\rceil\}}$

We notice that in the case of inequality constrained SOS1, we can add a slack variable s to transform it into an equality constrained SOS1. Applying the last result we can model an inequality constrained SOS1 with $\lceil\log_2(|I|+1)\rceil$ extra binary variables and constraints and relax the integrality of the original variables.

4.2.2 Implicit branching of the extended formulation

Implicitly, the introduction of these new variables and constraints allows the use of *constraint branchings* on whole sets of variables instead of branching on variables.

Single variable branching. Given some fractional solution $\bar{x}$, classical branching methods choose some fractional variable $\bar{x}_i$ and divide the current Branch and Bound node into two new branches where $x_i \leqslant \lfloor\bar{x}_i\rfloor = 0$ and $x_i \geqslant \lceil\bar{x}_i\rceil = 1$ respectively (see figure 4.2).

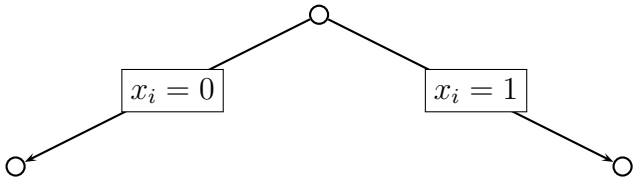


Figure 4.2: Classic variable branching

Traditional SOS1 Branching. In presence of SOS1 constraints, a classic approach ([13]) specialized for SOS1 determines some index t separating at least two fractional variables x_i and x_j such that $i \leqslant t \leqslant j$ and then branches over the disjunction $(x_p \leqslant 0, \forall p \leqslant t) \vee (x_p \leqslant 0, \forall p \geqslant t+1)$ (See figure 4.3). When doing a depth first search in this branching tree, we will branch at most $\lceil\log_2|I|\rceil$ times. Nevertheless, it can lead to unbalanced trees because of its dependence in t at each disjunction.

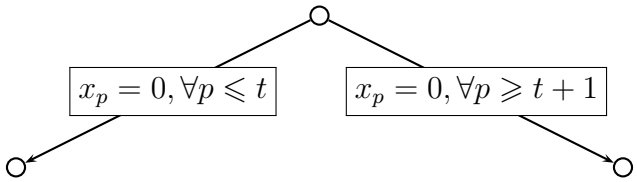


Figure 4.3: SOS1 branching

Independent SOS1 Branching. In the case of the extended formulation of Proposition 1, the underlying branching done has obviously the same *depth* of $\lceil \log_2 |I| \rceil$ (the number of extra variables). The main difference is that in order to reach any leaf of the tree, we have to branch over the same separations in order to reach any leaf of the tree (in general in a different order).

Given a fractional solution $\bar{x}$, the constraints (4.7) ensure that there exists some extra variable $\bar{z}_l$ which is fractional as well and we can branch on it, generating the disjunction $(z_l = 0) \vee (z_l = 1)$. The interpretation in terms of the original variables is the following: Given that $\bar{z}_l$ is fractional then the extra constraints (4.7) imply that we have simultaneously:

$$\sum_{p \in S^+_{|I|}(l)} \bar{x}_p > 0 \tag{4.10}$$

$$\sum_{p \in S^-_{|I|}(l)} \bar{x}_p > 0 \tag{4.11}$$

This way, branching on z_l separates fractional variables x with the branching depicted in figure 4.4: During a Branch and Bound, there is a lot of effort put into the selection of

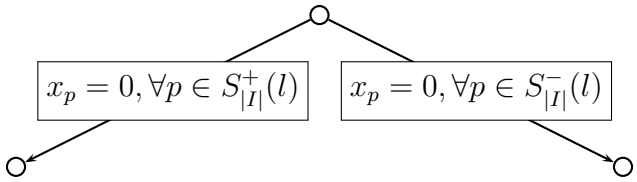


Figure 4.4: Constraint branching

the variable we branch on and in the selection of the branch we take to go on with the search. So an advantage of this formulation is that we have smaller Branch and Bound trees - because there are less integer variables - and that we let the solver decide the branch and node selections as usual. However, it leads to bigger formulations and eventually breaks any structure the original problem had, turning the LP resolutions heavier. One of the purposes of the next subsections is to detail the extended formulations and the underlying branching schemes for the original formulation so that we can compare their practical behavior in a future work.

4.2.3 Explicit branching for the original formulation

In [79], they show that the extended formulation is as strong as the original formulation, i.e.: that both LP relaxations of the formulations have the same set of feasible points. This

remark is of great importance: The reformulation's strength lies in its reduced number of integer variables but on another hand, the LP relaxations we have to solve are bigger in terms of number of variables and constraints. Eventually, any structure the original problem had can be altered by adding these additional constraints. In consequence, when solving our problem via Branch and Bound the extended LP relaxations we solve are more of a burden than the original ones. We propose a mixed approach that seeks to use the best of both formulations. The idea is to use a branching rule suggested by the extended formulation but to solve the LP relaxations of the original problem.

From the equivalence we showed in the last section, we can derive a constraint branching scheme for the original formulation. Indeed, when relaxing the integrality of all the variables we can always find a feasible fractional value of $\bar{z}$ given a fractional $\bar{x}$ from equalities (4.7). Actually, equations (4.7) explicitly show that we have a bijection between the values of the variables z and x. From this last observation we can deduce that if we had solved the extended formulation, the LP relaxation would have returned $\bar{z}_l = \sum_{p \in S^+_{|I|}(l)} \bar{x}_p$. Moreover, if the optimal LP solution $\bar{x}$ is fractional, the disjunctive constraint implies that there are at least two variables $\bar{x}_i$ and $\bar{x}_j$ that are fractional. Consequently, there exists some fractional $\bar{z}_l$ as well and as such, we can branch on the entire partition $\left\{S^+_{|I|}(l), S^-_{|I|}(l)\right\}$. In the next subsection, we will define a general methodology to solve difficult combinatorial problems on networks.

4.3 Application to simple path and Hamiltonian circuit

In the following, we present how the formulations presented above adapt to two network problems where we can always partition the variables such that the variables of each subset are bound by a single disjunctive constraint.

4.3.1 Simple path set

For every node $v \in V$ let give an index to each of its outgoing edges: $\delta^+(v) := \left\{e^v_1, ..., e^v_{|\delta^+(v)|}\right\}$. For the problems whose formulation implies that each solution must be a simple path - i.e.: that does not contain any cycle - we can remark that if the path goes through some node $v \in V$, then it does it only once and as such, the following implicit disjunctive constraints are valid:

$$\sum_{i=1}^{\left|\delta^+(v)\right|} x_{e^v_i} \leqslant 1 \quad \forall v \in V \tag{4.12}$$

For each node $v \in V$ we add a slack variable $x_{e_0^v}$ corresponding to a dummy edge e_0^v. This way we obtain the following equality constrained SOS1:

$$\sum_{i=0}^{\left|\delta^+(v)\right|} x_{e_i^v} = 1 \quad \forall v \in V \tag{4.13}$$

Extended formulation for simple paths problems Given that each node $v \in V$ defines a disjunctive constraint, we can apply the result of proposition 1 and replace the integrality of each $x_e \in \{0,1\}, \forall e \in \delta^+(v)$ by introducing new variables z_l^v and the following constraints:

$$\sum_{p \in S^+_{|\delta^+(v)|+1}(l)} x_{e_p^v} = z_l^v \quad \forall l \in \{1, ..., L(|\delta^+(v)|+1)\} \tag{4.14}$$

$$z^v \in \{0,1\}^{L(|\delta^+(v)|)} \tag{4.15}$$

We remark that using the sets of ingoing edges instead of the outgoing edges sets to cover the original variables is perfectly fine as well. Repeating the process for every node of G we obtain the following formulation:

$$\min_{x_e, z_l^v} \quad \rho(x) \tag{4.16}$$

$$\text{s.t.: } x \in X \tag{4.17}$$

$$\sum_{i=0}^{|\delta^+(v)|} x_{e_i^v} = 1 \qquad \forall v \in V \tag{4.18}$$

$$\sum_{p \in S^+_{|\delta^+(v)|+1}(l)} x_{e_p^v} = z_l^v \qquad \forall v \in V, \forall l \in \{1, ..., L(|\delta^+(v)|+1)\} \tag{4.19}$$

$$z_l^v \in \{0,1\} \qquad \forall v \in V, \forall l \in \{1, ..., L(|\delta^+(v)|+1)\} \tag{4.20}$$

$$x_e \geqslant 0 \qquad \forall e \in E \tag{4.21}$$

This extended formulation has $\sum_{v \in V} \lceil \log_2(|\delta^+(v)|+1) \rceil$ zero-one variables, which can be upper bounded by:

$$\begin{aligned}
\sum_{v \in V} \lceil \log_2(|\delta^+(v)|+1) \rceil &\leqslant \max_{\alpha \in \mathbb{R}^n} \left\{ \sum_{v \in V} \lceil \log_2(\alpha_v + 1) \rceil : \sum_{v \in V} \alpha_v \leqslant m \right\} \\
&< \max_{\alpha \in \mathbb{R}^n} \left\{ n + \sum_{v \in V} \log_2(\alpha_v + 1) : \sum_{v \in V} \alpha_v \leqslant m \right\} \\
&= n + n \log_2 \left(\frac{m}{n} + 1 \right)
\end{aligned}$$

We can show that this bound is asymptotically tight with the following family of instances: Let K_n be the complete directed graph with $n = 2^q + 1$. Given that K_n is complete we have

$m = n(n-1)$ and for each node $v \in V$ we have that $|\delta^+(v)| = n-1$. Consequently:

$$\frac{n\left(1+\log_2(\frac{m}{n}+1)\right)}{\sum_{i=1}^{n}\lceil\log_2(|\delta^+(i)|+1)\rceil} = \frac{1+\log_2(2^q+1)}{\lceil\log_2(2^q+1)\rceil}$$

$$= \frac{1+q+\log_2(1+\frac{1}{2^q})}{1+q} \underset{q\to+\infty}{\longrightarrow} 1$$

Putting everything together, the extended formulation has:

$$\begin{aligned} O(m) &\text{ continuous variables} \\ O\left(n\log_2\frac{m}{n}\right) &\text{ binary variables} \\ O\left(n\log_2\frac{m}{n}\right) &\text{ constraints} \end{aligned}$$

In the worst case we have $m = n(n-1)$ and hence the maximal size of a search tree drops from 2^m in the original case to $O\,(2n)^n$ with the extended formulation.

Although it reduces the number of integer variables, the extended formulation still has more variables and constraint than the original one. In consequence the LP relaxations can be slower and last but not least, destroy the good structure we had with the path polyhedron alone. For example, if we solve the relaxed problem with interior point methods or gradient descent algorithms, the iterations are very fast thanks to the special structure the problem has, using shortest path algorithm at every descent direction lookup. In this goal it can be hypothetically faster to use directly the underlying constraint branching that the extended formulation simulates. We can point out that this formulation can be beneficial for highly connected graphs, when the degrees of nodes are large and comparable to n. If the graph has bounded degree, then the reduction in integer variables is really minor. As in the last section, if at some node of the Branch and Bound tree the relaxed solution is fractional then we know that there is at least one of the nodes $v \in V$ of the network that has two outgoing edges e_i^v and e_j^v having fractional flows. Choosing the node with most fractional outgoing edges variables could be a first natural approach. In consequence, we can branch over the SOS1 constraint corresponding to the node v, $\sum_{p=1}^{|\delta^+(v)|} x_{e_p^v} \leqslant 1$ and we already know that there exists some $l \in \{1, ..., L\,(|\delta^+(v)|)\}$ such that we can separate the fractional variables $x_{e_i^v}$ and $x_{e_j^v}$ as depicted in figure 4.5.

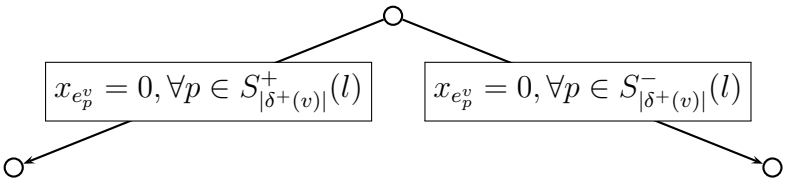


Figure 4.5: Constraint branching for the simple path set

A natural choice of the partitioning index l could be to select the one separating the most the fractional variables of the outgoing edges of v (i.e.:half of them in each subset of the partition $\left\{S^+_{|\delta^+(v)|}(l), S^-_{|\delta^+(v)|}(l)\right\}$). This way we ensure that the Branch and Bound tree

remains balanced using this constraint branching. We notice that for simple path problems, the extra slack variable for each node is unnecessary as the structure of the problem ensures that the solution is integer if and only if there is at most one outgoing edge used by the solution.

Cycle cancelling The typical path set formulation allows the formation of cycles and although mathematically correct, it is a clearly undesirable feature for a lot of routing problems. For example, in [92] they model the risk aversion of a *st*-traveler constraining its path to stochastically dominate a benchmark path. They prove that when the random variables describing the travel times are uncorrelated, their model returns cycle-free paths but in the general case the structure of this problem drives the model to return paths that could have cycles. Our constraint branching has the nice ability to deal with cycle formation whenever they are not isolated, i.e.: if the graph induced by the solution is connected. We can notice that if a cycle is entangled with the path, then there is at least one node crossed at least twice by the solution. Although the current node solution is integral, we can branch in the same way as before choosing a partition separating the two or more outcoming edges of some node crossed several times by the solution. However, such an approach will not cut isolated cycles.

4.3.2 Direct application: Traveling salesman problem

There exists an IP formulation for the TSP due to [74] that is formulated as a shortest path in a derived acyclic digraph with disjunctive side constraints:

$$\min_{x_e, x_e^k} \sum_{e \in E} c_e x_e \tag{4.22}$$

$$\text{s.t.: } x_e = \sum_{k=1}^{n} x_e^k \qquad \forall e \in E \tag{4.23}$$

$$\sum_{e \in \delta^-(i)} x_e^k - \sum_{e \in \delta^+(i)} x_e^{k+1} = 0 \qquad \forall i \in V \setminus \{1\}, \forall k \in \{2, ..., n-1\} \tag{4.24}$$

$$\sum_{e \in \delta^+(1)} x_e^1 = 1 \tag{4.25}$$

$$\sum_{e \in \delta^-(1)} x_e^n = 1 \tag{4.26}$$

$$\sum_{e \in \delta^-(i)} x_e = 1 \qquad \forall i \in V \tag{4.27}$$

$$\sum_{e \in \delta^+(i)} x_e = 1 \qquad \forall i \in V \tag{4.28}$$

$$x_e^k \geqslant 0 \qquad \forall k \in \{1, ..., n\}, \forall e \in E \tag{4.29}$$

$$x_e \in \{0, 1\} \qquad \forall e \in E \tag{4.30}$$

Where the variable x_e^k indicates if the edge e is in position k in the tour or not, and x_e if the edge is taken by the tour or not. The variables x_e^k are defined as continuous as their integrality

is enforced by the x_e. In this case, we can directly use the extended formulation with equalities described in the last section using either the constraints (4.28) or the constraints (4.27) as the disjunctive constraints to branch on. In either case, the extended formulation associated with this branching has

$$\begin{aligned} O(mn) &\text{ continuous variables} \\ O\left(n \log_2 \frac{m}{n}\right) &\text{ binary variables} \\ O\left(n^2\right) &\text{ constraints} \end{aligned}$$

To the best of our knowledge, this is the first formulation for the TSP with so few binary variables. Nevertheless, even solving the LP relaxation of the original formulation is expected to be challenging with $O(mn)$ variables and $O(n^2)$ constraints.

4.4 New formulations for hard problems in DAGs

In the last section, we saw that we could cover all the integer variables by disjunctive constraints in the case of the simple path set. In the case of DAGs, we will show that we can build a different type of covering for the path set and that a similar result can be found for the *st*-cut set.

4.4.1 Path set in DAGs

A number of difficult problems involve paths in DAGs: For example, given a fractional solution $\bar{x}$ of problem (4.1) at some node of the Branch and Bound, we can heuristically find a feasible solution looking for the best path contained in the graph induced by $\bar{x}$, $G(\bar{x}) := (V, \{e \in E : \bar{x}_e > 0\})$. For the type of problems we are interested in, we assume that any optimal solution is cycle-free. Consequently, $G(\bar{x})$ is a DAG and is smaller than the original network. In another context, the project management problem with resource constraints ([32]) consists in finding a longest path in a DAG subject to side constraints and has been proved to be NP hard.

We will assume that all the nodes of G are reachable from the source node s and that the sink node t is reachable from every node. If it is not the case, we can trivially preprocess all the nodes not connected to t or from s. In what follows we consider *st*-cuts with a particular structure. A cut is an edge set $C \subseteq E$ such that there is no *st*-path in $(V, E \setminus C)$. In particular we are interested in cuts C such that they cut every path in a DAG only once. In consequence, given any such cut C the following disjunctive constraint is a valid equality for the simple *st*-path set:

$$\sum_{e \in C} x_e = 1$$

This implies that whenever we obtain a fractional solution at some Branch and Bound node, there always exists one such cut $C \subseteq E$ such that at least two edges have fractional value. Indexing the edges of C as follows $C := \left\{e_1, ..., e_{|C|}\right\}$, there always exists some $l \in \{1, ..., L(|C|)\}$

such that we can branch as depicted in figure 4.6. Now, we want to find a set of such cuts

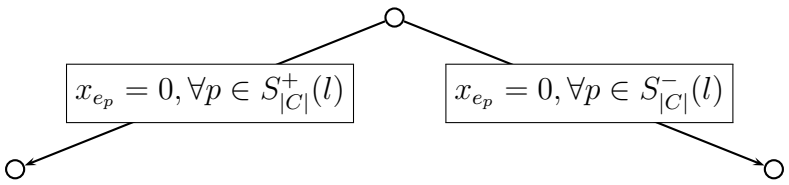


Figure 4.6: Constraint branching for the path set in a DAG

$\{C_1, ..., C_K\}$ such that the number of extra variables in the extended formulation is minimized. We first define the notion of *st*-blocking cut and introduce a way to compute the best edge covering with such cuts solving a MIP. Second, we propose a fast method to find an edge covering candidate and we then show a fast way to dynamically compute the best cut.

Definition: A set $C \subseteq E$ is an *st*-blocking cut for a DAG if there exists a set $Y \subseteq [0,1]^m \times \{0,1\}^n$ described below and $e \in C$ if and only if $y_e = 1$.

$$(y, \pi) \in Y \Leftrightarrow \begin{cases} y_{ij} = \pi_i - \pi_j & \forall (i,j) \in E \\ \pi_s - \pi_t = 1 & \\ y_e \in [0,1] & \forall e \in E \\ \pi_i \in \{0,1\} & \forall i \in V \end{cases} \tag{4.31}$$

Where y_e represents if edge $e \in E$ is in the *st*-blocking cut $C \subseteq E$ or not and π_i is one if $i \in V$ is reachable from s in $(V, E \backslash C)$ and π_i is zero if $i \in V$ can reach t in $(V, Ei \backslash C)$. Let prove that Y defines a cut: If $(y, \pi) \in Y$, let prove that there is no *st*-path in $(V, E \setminus \{e \in E : y_e = 1\})$. By integrality of π we can notice that we necessarily have $\pi_s = 1$ and $\pi_t = 0$, so given any *st*-path P there exists some edge $(i,j) \in P$ such that $y_{ij} = \pi_i - \pi_j = 1 - 0 = 1$. In other words, any *st*-path is cut by the set $\{e \in E : y_e = 1\}$. Moreover, the positivity of y variables implies that there are no edges with a tail $\pi_i = 0$ and head $\pi_j = 1$. Therefore it is impossible for an *st*-path P to intersect more than once the cut characterized by the set Y. In a graph without incoming arcs on s, e.g. a DAG, there is always one such *st*-blocking cut : letting $\pi_s = 1$ and all other $\pi_i = 0$ for all $i \in V \setminus \{s\}$. In what follows we formulate a problem to identify in a DAG a set of such blocking *st*-cuts. We notice that having such a *st*-blocking cut C at hand defines an equality constrained SOS1 constraint. The reformulation of proposition 1 allows to relax the integrality of the variables y_e for any $e \in C$ adding $(\lceil \log_2 |C| \rceil)$ extra binary variables and constraints. This problem aims to partition the set of edges of the graph into different *st*-blocking cuts with the fewer number of extra binary variables induced by proposition 1 for each blocking *st*-cut.

An optimal edge cover by *st*-blocking cuts. Lets define $\bar{G} := (V \cup \{\bar{s}\}, E \cup \{(\bar{s}, s)\})$ with $\bar{s}$ an additional node linked only to s. Let define $\bar{Y}$ as the homologue in $\bar{G}$ of Y in G. Consequently the optimal decomposition of E in terms of number of binary variables of the

extended formulation can be found solving the following problem:

$$\min_{y_e^k,\pi_i^k,t_e} \sum_{k=1}^{m} \left\lceil \log_2 \sum_{e\in E} y_e^k \right\rceil + \sum_{e\in E} t_e \tag{4.32}$$

$$\text{s.t.: } t_e + \sum_{k=1}^{m} y_e^k \geqslant 1 \qquad \forall e \in E \tag{4.33}$$

$$(y^k, \pi^k) \in \bar{Y} \qquad \forall k \in \{1, ..., m\} \tag{4.34}$$

$$t_e \geqslant 0 \qquad \forall e \in E \tag{4.35}$$

Where the variables (y^k, π^k) represent the k-th cut, and trivially we cannot have more than m cuts. The number of extra binary variables induced by the k-th cut is $\left\lceil \log_2 \sum_{e\in E} y_e^k \right\rceil$. Each variable t_e is one if edge e is not present in any cut and then count as a single stray variable. The extra edge we added in $\bar{G}$ represents the fact that if it is not convenient for some of the m st-blocking cuts to be used then it contains only this extra edge and has zero contribution in the objective value. This last problem is a nonlinear, non convex integer problem. Nevertheless, for any integer $r > 1$ and any number $\lambda \in \mathbb{R}$, we can notice that we have $\lceil \log_r \lambda \rceil = \min \left\{ l \in \mathbb{N} : r^l \geqslant \lambda \right\}$ which we can rewrite as follows:

$$\lceil \log_r \lambda \rceil = \min_{b_l} \sum_{l=0}^{L} l b_l \tag{4.36}$$

$$\text{s.t.: } \sum_{l=0}^{L} r^l b_l \geqslant \lambda \tag{4.37}$$

$$\sum_{l=0}^{L} b_l = 1 \tag{4.38}$$

$$b \in \{0,1\}^L \tag{4.39}$$

With L an a priori upper bound of $\lceil \log_r q \rceil$. Putting everything together, we can find the optimal edge covering with st-blocking cuts solving the following MIP (4.4.1)

$$\min_{b_l^k, y_e^k, \pi_i^k, t_e} \sum_{k=1}^{m} \sum_{l=0}^{L(m)} l b_l^k + \sum_{e\in E} t_e \tag{4.40}$$

$$\text{s.t.: } (y^k, \pi^k) \in \bar{Y} \qquad \forall k \in \{1, ..., m\} \tag{4.41}$$

$$t_e + \sum_{k=1}^{m} y_e^k \geqslant 1 \qquad \forall e \in E \tag{4.42}$$

$$\sum_{l=0}^{L(m)} 2^l b_l^k \geqslant \sum_{e\in E} y_e^k \qquad \forall k \in \{1, ..., m\} \tag{4.43}$$

$$\sum_{l=0}^{L(m)} b_l^k = 1 \qquad \forall k \in \{1, ..., m\} \tag{4.44}$$

$$b^k \in \{0,1\}^{L(m)} \qquad \forall k \in \{1, ..., m\} \tag{4.45}$$

$$t_e \geqslant 0 \qquad \forall e \in E \tag{4.46}$$

which is likely harder than the original problem in general, but can be useful for method comparison purposes.

A practical decomposition for DAGs. Now we will present a practical way to build an a priori edge covering with *st*-blocking cuts. A layered directed graph is a graph whose nodes are partitioned into several layers such that there is no edge linking two nodes of a same layer. In [30], they show that we can always draw a layered graph representation of any DAG in linear time. Further, they show that the resulting layered graph is the one with the least number of layers. To draw such a representation, they first build the tree of longest distances in G, which in the case of DAGs can be done in $O(m)$ with any search algorithm (but is NP-hard for general graphs). Second, they build the layers grouping the vertices having the same distance from s with respect to the number of edges (see figure 4.7). From this transformation

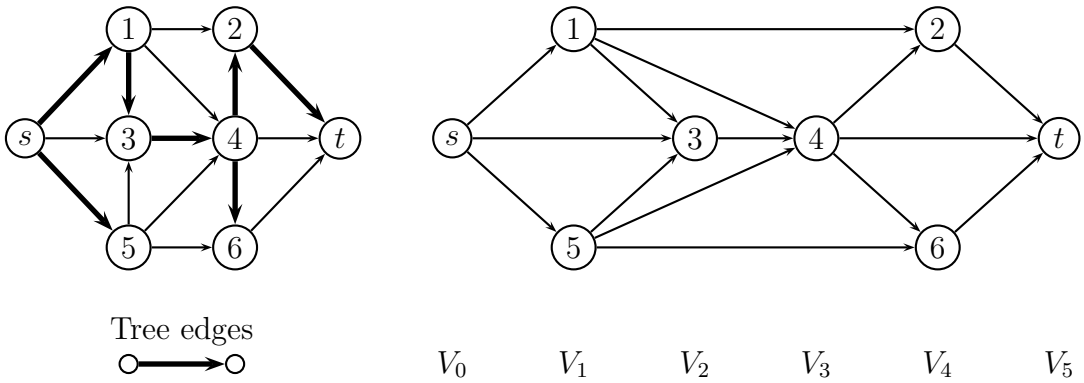


Figure 4.7: Layer representation of a DAG from longest distance tree

we can find a very natural edge cover: Defining V_k as the set of nodes at a longest distance of k from s, each set $C_k := \{(i,j) \in E : \exists p \leqslant k-1, q \geqslant k, i \in V_p, j \in V_q\}$ defines a blocking *st*-cut. As indicated earlier, this edge cover $\{C_1, ..., C_d\}$ has at most d layers, with d the length of the longest path in G. We can directly replace the SOS1 constraints but we could potentially have some variables in several blocking cuts. To tackle this, we notice that for any set of indices defining an equality constrained SOS1, any subset defines a inequality constrained SOS1. Adding a slack variable we get an equality constrained SOS1 and we can reformulate it with a logarithmic number of extra binary variables and constraints. In our case, we will take the sets $\bar{C}_k := \{(i,j) \in \delta^+(V_k)\} \subseteq C_k$ to cover all the original integer variables. This edge cover $\{\bar{C}_1, ..., \bar{C}_d\}$ is also a partition of E and as in the simple path case, we can easily show that the total number of extra binary variables and constraints of the reformulation is at most $d\left(1 + \log_2\left(\frac{m}{d} + 1\right)\right)$.

On the fly disjunction generation Taking the original formulation of (4.1) in a DAG, at each node of the Branch and Bound tree we can generate the best valid set of variables separating the maximum number of fractional variables. To do so, given a fractional *st* 1-flow we want to find the *st*-blocking cut containing the maximum number of edges with fractional flow and then partition in two the edges of this *st*-blocking cut by assigning half of the fractional variables to each part in order top have a balanced disjunction. In the special case of DAGs, finding the maximum capacity *st*-blocking cut proved to be easily solvable.

In [25] they show that the maximum st-blocking cut problem in DAGs is in fact the dual of finding a minimum feasible flow in a capacitated network. Given a fractional solution $\bar{x}$, the problem of finding a st-blocking cut maximizing the number of fractional edges it contains can be written as follows:

$$\max_{y,\pi} \sum_{e \in E} \lceil \bar{x}_e \rceil y_e \tag{4.47}$$

$$\text{s.t.: } y_{ij} \leqslant \pi_i - \pi_j \qquad \forall (i,j) \in E \tag{4.48}$$

$$\pi_s - \pi_t \leqslant 1 \tag{4.49}$$

$$y_e \geqslant 0 \qquad \forall e \in E \tag{4.50}$$

$$\pi_i \geqslant 0 \qquad \forall i \in V \tag{4.51}$$

Notice that in the case of DAGs, we can relax the integrality of all the variables. Now we can easily show that its dual problem is:

$$\min_{\phi,f} f \tag{4.52}$$

$$\text{s.t.: } \sum_{e \in \delta^+(i)} \phi_e - \sum_{e \in \delta^-(i)} \phi_e \leqslant \begin{cases} f & \text{If } i = s \\ 0 & \forall i \in V \setminus \{s,t\} \\ -f & \text{If } i = t \end{cases} \tag{4.53}$$

$$\phi_e \geqslant \lceil \bar{x}_e \rceil \qquad \forall e \in E \tag{4.54}$$

$$f \geqslant 0 \tag{4.55}$$

Which is equivalent to a minimum cost flow problem and hence can be solved by several algorithms ([3]) in $O\left(n(m + n \log_2 n)\right)$ time. Next, we can retrieve the maximum cut from the minimum flow in almost the same way that we retrieve a minimum cut from a maximum flow: we run a search starting from s considering that we cannot pass trough edges where the minimum flow is equal to its lower bound on this edge. The maximum cut is the set of outgoing edges of the explored node set from s (see figure 4.8 where we assumed that all the edges had fractional $\bar{x}_e$). Once we have the st-blocking cut $C :=$

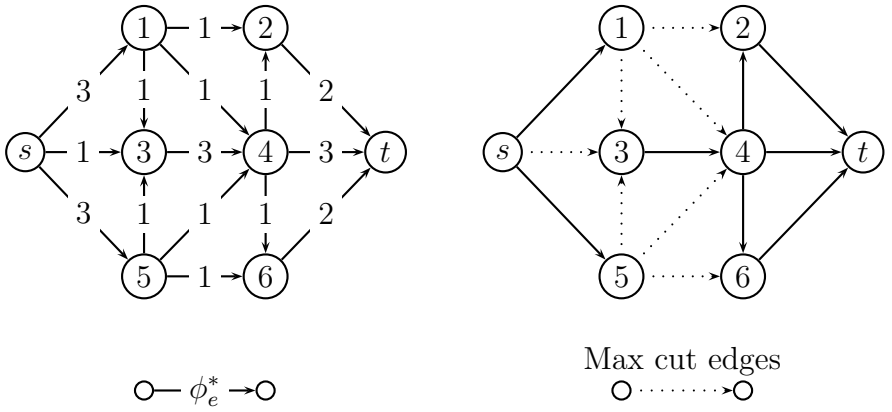


Figure 4.8: Maximum st-cut from minimum st-flow

$\{e_1, ..., e_{|C|}\}$ containing the maximum number of fractional edges and we branch on a disjunction $\left(x_{e_{[i]}} = 0, \forall i \in \left\{1, ..., \left\lceil \frac{|C|}{2} \right\rceil \right\}\right) \vee \left(x_{e_{[i]}} = 0, \forall i \in \left\{\left\lceil \frac{|C|}{2} \right\rceil + 1, ..., |C|\right\}\right)$ having half of

the fractional variables on each side. This way we ensure that we have a balanced tree, and we are branching on the biggest disjunction of fractional variables, then trying to minimize the depth of the tree. We could not find any bound for this methodology but we expect it to be more efficient than the other approaches presented in this chapter. In the next section, we will consider the same approach applied to the *st*-blocking cut set, where an interesting duality with this section appears in several results.

4.4.2 Cut set in DAGs

In this section, we will adapt the optimal covering method for the *st*-blocking cut set in DAGs and use in a dual way the other methods of the last section. This approach could help to solve more efficiently hard problems like resource constrained open pit mining scheduling problems ([21]) which can be formulated as the problem of finding a minimum *st*-cut in a DAG subject to side constraints.

As we noticed earlier, any *st*-blocking cut in a DAG destroys exactly one edge of any *st*-path. Consequently, for any *st*-path $P \subseteq E$ and any *st*-blocking cut defined by $(y, \pi) \in Y$ we have the following disjunctive constraint:

$$\sum_{e \in P} y_e = 1$$

As in the last section, we propose two ways to build an a priori edge cover of E with *st*-paths: We first show an optimal decomposition solving a MIP and then show how to quickly find a valid edge covering with *st*-paths. A dynamic search for the best covering at each Branch and Bound node is also presented.

An optimal edge cover by *st*-paths. Defining the graph $\widetilde{G} = (V, E \cup \{(s,t)\})$ and $\widetilde{X}$ the homologue in $\widetilde{G}$ of X in G, we can use the same trick for computing logarithms as in the last section, and then the optimal decomposition of E with *st*-paths in terms of number of

potential enumerations can be found solving the following MIP:

$$\min_{b_l^k, x_e^k t_e} \sum_{k=1}^{m} \sum_{l=0}^{L(m)} l b_l^k + \sum_{e \in E} t_e \tag{4.56}$$

$$\text{s.t.: } x^k \in \widetilde{X} \qquad \forall k \in \{1, ..., m\} \tag{4.57}$$

$$t_e + \sum_{k=1}^{m} x_e^k \geqslant 1 \qquad \forall e \in E \tag{4.58}$$

$$\sum_{l=0}^{L(m)} 2^l b_l^k \geqslant \sum_{e \in E} x_e^k \qquad \forall k \in \{1, ..., m\} \tag{4.59}$$

$$\sum_{l=0}^{L(m)} b_l^k = 1 \qquad \forall k \in \{1, ..., m\} \tag{4.60}$$

$$b^k \in \{0, 1\}^{L(m)} \qquad \forall k \in \{1, ..., m\} \tag{4.61}$$

$$t_e \geqslant 0 \qquad \forall e \in E \tag{4.62}$$

Which can be harder than the original problem in general so we can just use a heuristic to determine some edge covering by *st*-paths but keeping it in mind for practical comparison purposes. Ironically, this kind of problem can see its number of integer variables reduced applying the approach of the last section.

A practical decomposition. In the last section, the problem of finding a minimum number of *st*-blocking cuts covering E needed the computation of the longest path in G. These two problems are actually dual to each other ([30]). In [25] they show that finding an edge cover with the minimum number of *st*-paths is in fact equivalent to find a minimum *st*-flow such that at least one unit of flow goes through each edge. Indeed, the amount of flow passing through each edge represents the number of covering *st*-paths crossing it, and the total flow f sent from s to t is the total number of covering *st*-paths. So in the end we just have to solve the minimum cost flow problem used in the last section to find a maximum cut, with an unitary lower bound on the flow of each edge:

$$\min_{f, \phi} \quad f \tag{4.63}$$

$$\text{s.t.: } \sum_{e \in \delta^+(i)} \phi_e - \sum_{e \in \delta^-(i)} \phi_e \leqslant \begin{cases} f & \text{If } i = s \\ 0 & \forall i \in V \setminus \{s, t\} \\ -f & \text{If } i = t \end{cases} \tag{4.64}$$

$$\phi_e \geqslant 1 \qquad \forall e \in E \tag{4.65}$$

$$f \geqslant 0 \tag{4.66}$$

Which can be done in $O\left(n(m + n \log_2 n)\right)$ using minimum cost flow algorithms ([3]). Now that we can cover all the edges of E with f *st*-paths $\{P_1, ..., P_f\}$ (with f the size of the maximum *st*-directed cut, as shown in the last section), we can directly reformulate each equality constrained SOS1 by its logarithmic reformulation. As in the *st*-path case, it is likely that $\{P_1, ..., P_f\}$ is not a partition of E as some edges can belong to several paths

of $\{P_1, ..., P_f\}$. To tackle this we can use the same trick as in the last section and define $\left\{\bar{P}_1, ..., \bar{P}_f\right\}$ as a partition of E such that $\bar{P}_k \subseteq P_k$ for every $k \in \{1, ..., f\}$. Such a partition is easy to find from $\{P_1, ..., P_f\}$ letting each edge belonging to exactly one $\bar{P}_k$ for some k. Adding a slack variable to each of the SOS1 defined by the sets $\bar{P}_k$, we can use the logarithmic reformulation and prove that the final problem has at most $f\left(1 + \log_2\left(\frac{m}{f} + 1\right)\right)$ extra binary variables and constraints.

On the fly disjunction generation Taking the original formulation, at each node of the Branch and Bound tree we can generate the best valid set of variables separating the maximum number of fractional variables. To do so, given a fractional *st* *st*-blocking cut we want to find the *st*-path containing the maximum number of edges with fractional value and then partition in two the edges of this path, assigning half of the fractional variables to each part in order top have a balanced disjunction. In the special case of DAGs, finding the longest path with respect to some weights can be done in linear time with any search algorithm. The longest path $P := \left\{e_1, ..., e_{|P|}\right\}$ with respect to weights $\left(\lceil \bar{x}_e \rceil\right)_{e \in E}$ will return the biggest disjunction, separating the maximum number of fractional variables as follows: $\left(y_{e_{[i]}} = 0, \forall i \in \left\{1, ..., \left\lceil \frac{|P|}{2} \right\rceil\right\}\right) \vee \left(y_{e_{[i]}} = 0, \forall i \in \left\{\left\lceil \frac{|P|}{2} \right\rceil + 1, ..., |P|\right\}\right)$

4.5 Conclusions

We built extended formulations for path problems and TSP having less variables than their original formulations and proposed several resolution schemes for cut or path problems in DAGs. In future work we shall see if these methods are competitive against state of the art solvers. In particular, we are particularly interested in the performance of the branching schemes involving DAGs. It could help to find feasible solutions at each Branch and Bound node of shortest path problems with convex and nondecreasing functions, and speed up the resolution of Resource Constrained Open Pit Mining scheduling problems. Moreover, we should investigate if we can build the associated node and branch selections in the context of constraint branching

Chapter 5

Risk averse Stackelberg security games

5.1 Introduction

Our last contribution was to introduce risk aversion in a special class of Stackelberg games ([81]) where a leader player moves first and then the followers decide their actions, maximizing their utility. In airport security or coast guard patrol, security forces - the *leader* or *defender* - have limited capacity to defend a finite set of targets against human adversaries - the *followers* or *attackers*. A Stackelberg security game ([43]) is defined as a game where the leader decides a mixed strategy to maximize its utility, taking into account that the follower will observe this strategy and in turn decide its action to maximize its utility. In this situation, it is crucial to use resources wisely to minimize the damage done to the targets. Hence, an accurate knowledge of the attackers' behavior is central. Standard models assume a perfectly rational attacker that maximizes its utility knowing the defense strategy ([63, 43]), or that can deviate from an small ε from the optimal attack ([64]). Nevertheless, it is commonly accepted that human decisions are in general different from the best policy to use ([18]). Consequently, assuming a highly intelligent adversary can lead to weak defense strategies, failing to take advantage of the attackers' known weaknesses. The work presented in [58] assumes that human adversaries do not behave rationally, sometimes selecting actions that do not maximize their utility. The model considered assumed attackers followed a Quantal Response Equilibria (QRE). This idea models the decision probability of an attacker with a logit based expression derived from discrete choice theory. Its parametrization with a *degree of rationality* contains, in fact, the perfect rationality or the indifference as special cases, and is strongly backed in the literature and in practice by its superior ability to model human behavior ([39, 72, 85, 90]). In [91] they solve in polynomial time the problem of finding an optimal - in expectation - defense strategy against quantal response adversaries in security games casting it as a continuous convex programming problem. In this work, we defined a natural extension of this expected utility maximization approach including risk aversion in the objective of the defender. We use a change of variables presented in [91] to polynomially find the best defense strategy when the defender wants to minimize an Entropic risk measure.

We show that the same method becomes useless when extending our latter model to the case where there are several types of attacker with different rationality degrees and different impacts on the payoffs of the defender. Nevertheless, we show how find an ε-approximation of the problem solving convex MINLP.

We structured the rest of the chapter as follows: in the next section we present the result of [91] which solves the security game with risk neutral defender. In section 5.3 we prove that we can solve polynomially the problem when introducing risk aversion. Section 5.4 shows how to formulate as a convex integer programming problem the game with risk aversion when facing several types of adversary. We show in section 5.5 how to quickly evaluate the payoffs probability distributions of the defender without sampling. We present our conclusions in section 5.6.

5.2 Quantal response equilibria in security games

We first consider a Stackelberg security game with a single leader (defender) maximizing its expected utility and a single attacker ([43]) following a quantal response (QR) as was considered in [91]. If the attacker targets place $i \in \{1, ..., n\}$ and the defender blocks the attack, then the reward of the defender is $\bar{R}_i \geqslant 0$ and the penalty of the attacker is $P_i \leqslant 0$. On the other hand, if there is an attack on an undefended target $i \in \{1, ..., n\}$, the defender receives a penalty $\bar{P}_i \leqslant 0$ but the attacker obtains a reward $R_i \geqslant 0$. Taking the role of the defender we want to know how to maximize our utility using a total of $m < n$ resources to cover the n targets. Let $x_i \in [0, 1]$ be the frequency of protecting target i. It follows that the expected utility of the defender when the target i is attacked is:

$$\bar{U}_i(x_i) = x_i \bar{R}_i + (1 - x_i)\bar{P}_i \tag{5.1}$$

and the expected utility of the attacker when targeting place i is:

$$U_i(x_i) = x_i P_i + (1 - x_i)R_i \tag{5.2}$$

Assuming that the attacker is not perfectly rational and follows a quantal response of rationality factor $\lambda > 0$ ([58]), its probability to attack target i has probability $y_i(x)$:

$$y_i(x) = \frac{e^{\lambda U_i(x_i)}}{\sum_{j=1}^{n} e^{\lambda U_j(x_j)}} \tag{5.3}$$

We can see that perfect rationality ($\lambda = +\infty$) or indifference ($\lambda = 0$) of the adversary are special cases of the QR in equation (5.3). As the defender is trying to maximize its expected utility the problem to solve is then:

$$\max_{x \in [0,1]^n} \left\{ \sum_{i=1}^{n} y_i(x)\bar{U}_i(x_i) : \sum_{i=1}^{n} x_i \leqslant m \right\}$$

Defining: $\beta_i := e^{\lambda R_i} \geqslant 0$, $\gamma_i := \lambda(R_i - P_i) \geqslant 0$ and $\delta_i := \bar{R}_i - \bar{P}_i \geqslant 0$, We obtain:

$$\max_{x\in[0,1]^n} \left\{ \frac{\sum_{i=1}^{n} \beta_i e^{-\gamma_i x_i} \left(\bar{P}_i + \delta_i x_i\right)}{\sum_{i=1}^{n} \beta_i e^{-\gamma_i x_i}} : \sum_{i=1}^{n} x_i \leqslant m \right\} \tag{5.4}$$

Which is a highly nonlinear and non-convex optimization problem. We will now present the approach of [91] that solves polynomially problem (5.4).

Proposition 2. *Given two functions $N : X \subseteq \mathbb{R}^n \longmapsto \mathbb{R}$ and $D : X \subseteq \mathbb{R}^n \longmapsto \mathbb{R}^+ \setminus \{0\}$, and any $r \in \mathbb{R}$ we have:*

$$\max_{x\in X} \frac{N(x)}{D(x)} \leqslant r \Leftrightarrow \forall x \in X : N(x) - rD(x) \leqslant 0 \tag{5.5}$$

Proof. Let $x^* \in \arg\max_{x\in X} \frac{N(x)}{D(x)}$.

$\Rightarrow$: If $\frac{N(x^*)}{D(x^*)} \leqslant r$ by optimality of x^* we have: $\frac{N(\bar{x})}{D(\bar{x})} \leqslant r$ for any $\bar{x} \in X$

$\Leftarrow$: If $\frac{N(x^*)}{D(x^*)} > r$, then $\exists \bar{x} = x^* \in X : N(\bar{x}) - rD(\bar{x}) > 0$ □

The last proposition suggests the following scheme to solve approximately the optimization problem of equation (5.5):

Proposition 3. *Given a lower bound L and an upper bound U of $\max_{x\in X} \frac{N(x)}{D(x)}$, we can find an ε-optimal solution of the optimization problem (5.5) solving the following problem*

$$w(r) = \max_{x\in X} \{N(x) - rD(x)\} \tag{5.6}$$

with at most $\log_2 \frac{U-L}{\varepsilon}$ different values of r.

Proof. We first compute the value $w\left(r := \frac{U+L}{2}\right)$. If $w\left(\frac{U+L}{2}\right) > 0$, then by proposition 2 we can deduce that $w^* > r$ and we replace $L \leftarrow r$ else, if $w\left(\frac{U+L}{2}\right) \leqslant 0$, then by proposition 2 we can deduce that $w^* \leqslant r$ and we replace $U \leftarrow r$. We actualize $r \leftarrow \frac{U+L}{2}$ and we repeat the procedure until $U - L \leqslant \varepsilon$. At each step, we half the width of the interval so we reach the tolerance ε in at most $\log_2 \frac{U-L}{\varepsilon}$ steps. □

Let see how to apply this binary search scheme. The most important part is how to solve efficiently the subproblems at each step of the binary search.

Proposition 4. *Solving an iteration of the binary search is equivalent to solving the following problem:*

$$\max_{z} \quad -\sum_{i=1}^{n} \frac{\delta_i \beta_i}{\gamma_i} z_i \ln z_i + \sum_{i=1}^{n} \left(\bar{P}_i - r\right) \beta_i z_i \tag{5.7}$$

$$s.t.: \quad -\sum_{i=1}^{n} \frac{1}{\gamma_i} \ln z_i \leqslant m \tag{5.8}$$

$$z_i \in [e^{-\gamma_i}, 1] \qquad \forall i \in \{1, ..., n\} \tag{5.9}$$

Which is a concave maximization problem over a convex set.

Proof. First, the problem we have to solve at each iteration of the binary search is:

$$\max_{x} \quad \sum_{i=1}^{n} \beta_i e^{-\gamma_i x_i} \left(\bar{P}_i + \delta_i x_i\right) - r \sum_{i=1}^{n} \beta_i e^{-\gamma_i x_i} \tag{5.10}$$

$$\text{s.t.:} \quad \sum_{i=1}^{n} x_i \leqslant m \tag{5.11}$$

$$x_i \in [0,1] \qquad \forall i \in \{1,...,n\} \tag{5.12}$$

Introducing the following invertible change of variables: $z_i := e^{-\gamma_i x_i}$ (i.e.:$x_i := -\frac{1}{\gamma_i} \ln z_i$) the problem we have to solve can be rewritten as:

$$\max_{z} \quad -\sum_{i=1}^{n} \frac{\delta_i \beta_i}{\gamma_i} z_i \ln z_i + \sum_{i=1}^{n} \left(\bar{P}_i - r\right) \beta_i z_i \tag{5.13}$$

$$\text{s.t.:} \quad -\sum_{i=1}^{n} \frac{1}{\gamma_i} \ln z_i \leqslant m \tag{5.14}$$

$$z_i \in [e^{-\gamma_i}, 1] \qquad \forall i \in \{1,...,n\} \tag{5.15}$$

It is easy to show that the feasible set is convex thanks to the convexity of the functions $z_i \to -\frac{1}{\gamma_i} \ln z_i$. In the same way the objective function is concave given that the functions $z_i \to -\frac{\delta_i \beta_i}{\gamma_i} z_i \ln z_i$ are all concave. □

Proposition 5. *$L := \frac{\sum_{i=1}^{n} \beta_i e^{-\gamma_i \frac{m}{n}} \left(\bar{P}_i + \delta_i \frac{m}{n}\right)}{\sum_{i=1}^{n} \beta_i e^{-\gamma_i \frac{m}{n}}}$ and $U := \frac{\sum_{i=1}^{n} \beta_i e^{-\gamma_i} \left(\bar{P}_i + \delta_i\right)}{\sum_{i=1}^{n} \beta_i e^{-\gamma_i}}$ are respectively lower and upper bounds for the optimal value of problem* (5.4)

Proof. Because we are maximizing, the expected value of any feasible solution provides a lower bound. In particular, we obtain L evaluating the uniform strategy $x_i = \frac{m}{n}$. If we relax the resource constraint, the corresponding problem is a relaxation and then yields a greater optimal value. Further, we know that the optimal solution of this relaxation is to defend all the targets with frequency one, hence obtaining U. □

Using the last propositions, we can solve efficiently problem (5.4). We will now show that we can use this methodology to solve a generalization of (5.4) where the defender is risk averse and there are several types of adversary.

5.3 Risk averse defender

A natural extension of the last model is to assume that the defender is risk averse. Consequently, the defender must minimize the risk associated to have bad outcomes even if it can imply a lower expected payoff. In the following, we assume that the leader is risk averse and wants to minimize an entropic risk measure of parameter $\alpha > 0$. We define the entropic risk measure of parameter $\alpha > 0$ of a random variable X by $\alpha \ln \mathbb{E}\left[e^{\frac{X}{\alpha}}\right]$. With this definition at

hand the leader wants to solve the following optimization problem:

$$\min_{x\in[0,1]^n}\left\{\alpha\ln\left(\sum_{i=1}^{n}y_i(x)e^{-\frac{\bar{U}_i(x_i)}{\alpha}}\right):\sum_{i=1}^{n}x_i\leqslant m\right\}$$

We notice that the expected value maximization model of section 5.2 is a special case of the last problem as $\alpha\ln\mathbb{E}\left[e^{\frac{X}{\alpha}}\right]\underset{\alpha\to+\infty}{\longmapsto}\mathbb{E}[X]$. And given that $t\to\alpha\ln t$ is non decreasing, the general problem the defender solves is:

$$\min_{x\in[0,1]^n}\left\{\frac{\sum_{i=1}^{n}e^{\lambda U_i(x_i)}e^{-\frac{\bar{U}_i(x_i)}{\alpha}}}{\sum_{i=1}^{n}e^{\lambda U_i(x_i)}}:\sum_{i=1}^{n}x_i\leqslant m\right\}$$

Defining: $\beta_i:=e^{\lambda R_i}\geqslant 0$, $\gamma_i:=\lambda\left(R_i-P_i\right)\geqslant 0$ and $\mu_i:=e^{\lambda R_i-\alpha^{-1}\bar{P}_i}\geqslant 0$ and $\theta_i:=\lambda\left(R_i-P_i\right)+\alpha^{-1}\left(\bar{R}_i-\bar{P}_i\right)\geqslant 0$, we obtain:

$$\min_{x\in[0,1]^n}\left\{\frac{\sum_{i=1}^{n}\mu_i e^{-\theta_i x_i}}{\sum_{i=1}^{n}\beta_i e^{-\gamma_i x_i}}:\sum_{i=1}^{n}x_i\leqslant m\right\}\tag{5.16}$$

Proposition 6. *We can solve problem* (5.16) *using a binary search in* r *that solves at each iteration the following problem:*

$$w(r):=\min_{z}\quad\sum_{i=1}^{n}\mu_i z_i^{\frac{\theta_i}{\gamma_i}}-r\sum_{i=1}^{n}\beta_i z_i\tag{5.17}$$

$$s.t.:\qquad-\sum_{i=1}^{n}\frac{1}{\gamma_i}\ln z_i\leqslant m\tag{5.18}$$

$$z_i\in[e^{-\gamma_i},1]\qquad\forall i\in\{1,...,n\}\tag{5.19}$$

Which is a convex minimization problem.

Proof. Given that problem (5.16) is a fractional programming problem, we can solve it with binary search from proposition 3. At each iteration of the binary search, we have to solve the following problem:

$$\min_{x\in[0,1]^n}\left\{\sum_{i=1}^{n}\mu_i e^{-\theta_i x_i}-r\sum_{i=1}^{n}\beta_i e^{-\gamma_i x_i}:\sum_{i=1}^{n}x_i\leqslant m\right\}$$

Using the following invertible change of variables: $z_i:=e^{-\gamma_i x_i}$ (i.e.: $x_i:=-\frac{1}{\gamma_i}\ln z_i$), the problem we have to solve is:

$$\begin{aligned}\min_{z}\quad&\sum_{i=1}^{n}\mu_i z_i^{\frac{\theta_i}{\gamma_i}}-r\sum_{i=1}^{n}\beta_i z_i\\ \text{s.t.:}\quad&-\sum_{i=1}^{n}\frac{1}{\gamma_i}\ln z_i\leqslant m\\ &z_i\in[e^{-\gamma_i},1]\qquad\forall i\in\{1,...,n\}\end{aligned}$$

We already proved that the feasible set is convex and given that $\theta_i>\gamma_i$ the objective function is convex as well. □

Proposition 7. *$L := \frac{\sum_{i=1}^{n} \mu_i e^{-\theta_i}}{\sum_{i=1}^{n} \beta_i e^{-\gamma_i}}$ and $U := \frac{\sum_{i=1}^{n} \mu_i e^{-\theta_i \frac{m}{n}}}{\sum_{i=1}^{n} \beta_i e^{-\gamma_i \frac{m}{n}}}$ are respectively lower and upper bounds for the optimal value of problem* (5.16)

Proof. Similar to the proof of proposition 5.2. □

In the next section, we will show that we can solve the risk averse problem (5.16) when there is several types of adversary casting the problem as a convex MINLP.

5.4 Multiple types of adversary

In the following, we will assume that the attack can come from one of several types of adversary $a \in \{1, ..., A\}$ of respective penalties $P_i^a \leqslant 0$, rewards $R_i^a \geqslant 0$ and degrees of rationality λ_a. The reward and penalty of the defender when defending target i against an attacker of type a are respectively $\bar{R}_i^a$ and $\bar{P}_i^a$ and yield an utility $\bar{U}_i^a(x_i)$. We will assume a priori that the probability that the attack come from an adversary of type a is p_a, with $\sum_{a=1}^{A} p_a = 1$. The extended problem to solve is:

$$\min_{x \in [0,1]^n} \left\{ \alpha \ln \left(\sum_{a=1}^{A} p_a \sum_{i=1}^{n} y_i^a(x) e^{-\frac{\bar{U}_i^a(x_i)}{\alpha}} \right) : \sum_{i=1}^{n} x_i \leqslant m \right\}$$

As in section 5.3, let define: $\beta_i^a := e^{\lambda_a R_i^a} > 0$, $\gamma_i^a := \lambda_a(R_i^a - P_i^a) > 0$ and $\mu_i^a := e^{\lambda_a R_i^a - \alpha^{-1} \bar{P}_i^a} > 0$ and $\theta_i^a := \lambda_a(R_i^a - P_i^a) + \alpha^{-1}(\bar{R}_i^a - \bar{P}_i^a) > 0$, We then obtain:

$$\min_{x \in [0,1]^n} \left\{ \sum_{a=1}^{A} p_a \frac{\sum_{i=1}^{n} \mu_i^a e^{-\theta_i^a x_i}}{\sum_{i=1}^{n} \beta_i^a e^{-\gamma_i^a x_i}} : \sum_{i=1}^{n} x_i \leqslant m \right\} \tag{5.20}$$

Proposition 8. *The optimal solution of the following problem is an optimal solution for problem* (5.20)*.*

$$\min_{x,t,z} \quad \sum_{a=1}^{A} p_a \sum_{i=1}^{n} \mu_i^a e^{-\theta_i^a x_i - t_a} \tag{5.21}$$

$$s.t.: \quad \sum_{i=1}^{n} x_i \leqslant m \tag{5.22}$$

$$e^{t_a} \leqslant \sum_{i=1}^{n} \beta_i^a z_i^{\frac{\gamma_i^a}{\bar{\gamma}_i}} \qquad \forall a \in \{1, ..., A\} \tag{5.23}$$

$$z_i \leqslant e^{-\bar{\gamma}_i x_i} \qquad \forall i \in \{1, ..., n\} \tag{5.24}$$

$$x \in [0,1]^n \tag{5.25}$$

$$t \in \mathbb{R}^A \tag{5.26}$$

$$z_i \in [e^{-\bar{\gamma}_i}, 1] \qquad \forall i \in \{1, ..., n\} \tag{5.27}$$

Where $\bar{\gamma}_i := \max_{a \in \{1,...,A\}} \gamma_i^a$. *Constraints* (5.24) *are the only non-convex constraints.*

Proof. First, because the denominators of the objective function of problem (5.20) are strictly positive, let introduce new variables $t_a \in \mathbb{R}$ such that $e^{t_a} := \sum_{i=1}^{n} \beta_i^a e^{-\gamma_i^a x_i}$. Problem (5.20) is then equivalent to:

$$\begin{aligned} \min_{x \in [0,1]^n, t \in \mathbb{R}^A} \quad & \sum_{a=1}^{A} p_a \sum_{i=1}^{n} \mu_i^a e^{-\theta_i^a x_i - t_a} \\ \text{s.t.:} \quad & \sum_{i=1}^{n} x_i \leqslant m \\ & e^{t_a} \leqslant \sum_{i=1}^{n} \beta_i^a e^{-\gamma_i^a x_i} \qquad \forall a \in \{1, ..., A\} \end{aligned}$$

Which has a convex objective function, but a non-convex feasible set. Let $\bar{\gamma}_i := \max_{a \in \{1,...,A\}} \gamma_i^a$. Introducing new variables $z_i := e^{-\bar{\gamma}_i x_i}$ we can reformulate the last problem as follows:

$$\begin{aligned} \min_{x,t,z} \quad & \sum_{a=1}^{A} p_a \sum_{i=1}^{n} \mu_i^a e^{-\theta_i^a x_i - t_a} \\ \text{s.t.:} \quad & \sum_{i=1}^{n} x_i \leqslant m \\ & e^{t_a} \leqslant \sum_{i=1}^{n} \beta_i^a z_i^{\frac{\gamma_i^a}{\bar{\gamma}_i}} \qquad \forall a \in \{1, ..., A\} \\ & z_i \leqslant e^{-\bar{\gamma}_i x_i} \qquad \forall i \in \{1, ..., n\} \\ & x \in [0,1]^n \\ & t \in \mathbb{R}^A \\ & z_i \in [e^{-\bar{\gamma}_i}, 1] \qquad \forall i \in \{1, ..., n\} \end{aligned}$$

Given that $\bar{\gamma}_i \geqslant \gamma_i^a$, we can notice that the functions $z_i \longmapsto z_i^{\frac{\gamma_i^a}{\bar{\gamma}_i}}$ are concave, finishing the proof. □

To cast this last problem into something solvable, we will use the method described by [79] to piecewise linear approximate non-convex functions.

Proposition 9. *[79]. We can model any univariate function* $f : [l, u] \longmapsto \mathbb{R}$ *in the following*

way:

$$f(x) \approx \sum_{k=0}^{K} w^k f(d_k) \tag{5.28}$$

$$x = \sum_{k=0}^{K} w^k d_k \tag{5.29}$$

$$\sum_{k=0}^{K} w^k = 1 \tag{5.30}$$

$$\sum_{p \in S_K^+(l)} w^p \leqslant v^l \qquad \forall l \in \{1, ..., L(K)\} \tag{5.31}$$

$$\sum_{p \in S_K^-(l)} w^p \leqslant 1 - v^l \qquad \forall l \in \{1, ..., L(K)\} \tag{5.32}$$

$$v^l \in \{0, 1\} \qquad \forall l \in \{1, ..., L(K)\} \tag{5.33}$$

$$w^k \geqslant 0 \qquad \forall k \in \{1, ..., K\} \tag{5.34}$$

With $l = d_0 < d_1 < ... < d_K$ *a discretization of* $[l, u]$, $L(K) = \lceil \log_2 K \rceil$ *and for any* $l \in \{1, ..., L(K)\}$*:*

$$S_K^+(l) := \{p \in \{0, ..., K\} : \forall q \in Q_K(p), (B_K(q))_l = 1\} \tag{5.35}$$

$$S_K^-(l) := \{p \in \{0, ..., K\} : \forall q \in Q_K(p), (B_K(q))_l = 0\} \tag{5.36}$$

Where $Q_K(p) := \{q \in \{1, ..., K\} : p \in \{q - 1, q\}\}$ *and* $B_K : \{1, ..., K\} \longmapsto \{0, 1\}^{L(K)}$ *a bijective mapping such that for all* $q \in \{1, ..., K - 1\}$, $B_K(q)$ *and* $B_K(q + 1)$ *differ in at most one component (See reflected binary or Gray code in [38]). Such a gray code can be found quickly with the recursive algorithm of [45].*

Proof. This is only a sketch of the proof, as a detailed one can be found in [79] and is out of the scope of this work. The general idea here is to find with a logarithmic number of integer variables the interval $[d_k, d_{k+1}]$ where x lies and approximate $f(x) \approx w_k f(d_k) + w_{k+1} f(d_{k+1})$ where we have implicitly that $x = w_k d_k + w_{k+1} d_{k+1}$. A model of [13] tackles this so called *convex combination method* using type 2 Special Ordered Set constraints (SOS2). SOS2 constraints force a set of variables to sum up to one, with at most two of them with nonzero values and the extra requirement that the two nonzero variables must have adjacent indexes. The model used in this proposition simulates the SOS2 constraint of the convex combination method using a logarithmic number of extra variables and extra constraints to determine in which interval x belongs to. Let take the example of $K = 8$. One of the possible gray codes associated to the set of pairs $(\{q - 1, q\})_{q \in \{1,...,8\}}$ and the corresponding sets $(Q_K(p))_{p \in \{0,...,8\}}$ are shown is figure 5.1. The special structure of the sets $S_K^+(l)$ and $S_K^-(l)$ is illustrated with the same case where $K = 8$ in figure 5.2a. In figure 5.2a, the black cells represent the indices contained in each set, In figure 5.2b, we can see all possible combinations of $v \in \{0, 1\}^{L(K)}$ and their implications in terms of the SOS2 constrained original variables $w \in [0, 1]^{K+1}$. □

Proposition 10. *Defining* $\bar{\gamma}_i := \max_{a \in \{1,...,A\}} \gamma_i^a$ *and given a discretization in* K *intervals:*

$$\min_{i \in \{1,...,n\}} e^{-\bar{\gamma}_i} = d_0 < d_1 < ... < d_K = 0$$

q	$\{q-1,q\}$	$B_K(q)$		
1	$\{0,1\}$	0	0	0
2	$\{1,2\}$	0	0	1
3	$\{2,3\}$	0	1	1
4	$\{3,4\}$	0	1	0
5	$\{4,5\}$	1	1	0
6	$\{5,6\}$	1	1	1
7	$\{6,7\}$	1	0	1
8	$\{7,8\}$	1	0	0

(a) Example of Gray code for $K = 8$.

p	$Q_K(p)$
0	$\{1\}$
1	$\{1,2\}$
2	$\{2,3\}$
3	$\{3,4\}$
4	$\{4,5\}$
5	$\{5,6\}$
6	$\{6,7\}$
7	$\{7,8\}$
8	$\{8\}$

(b) Sets $(Q_K(p))_{p\in\{0,...,K\}}$ for $K = 8$

Figure 5.1: Example of Gray code and sets $(Q_K(p))_{p\in\{0,...,K\}}$ for $K = 8$

we can approximately solve problem (5.20) *by solving the following convex MINLP:*

$$\min_{x,t,z,v,w} \quad \sum_{a=1}^{A} p_a \sum_{i=1}^{n} \mu_i^a e^{-\theta_i^a x_i - t_a} \tag{5.37}$$

$$s.t.: \quad \sum_{i=1}^{n} x_i \leqslant m \tag{5.38}$$

$$e^{t_a} \leqslant \sum_{i=1}^{n} \beta_i^a z_i^{\frac{\gamma_i^a}{\bar{\gamma}_i}} \qquad \forall a \in \{1,...,A\} \tag{5.39}$$

$$z_i \leqslant \sum_{k=0}^{K} w_i^k e^{d_k} \qquad \forall i \in \{1,...,n\} \tag{5.40}$$

$$-\bar{\gamma}_i x_i = \sum_{k=0}^{K} w_i^k d_k \qquad \forall i \in \{1,...,n\} \tag{5.41}$$

$$\sum_{k=0}^{K} w_i^k = 1 \qquad \forall i \in \{1,...,n\} \tag{5.42}$$

$$\sum_{p\in S_K^+(l)} w_i^p \leqslant v_i^l \qquad \forall i \in \{1,...,n\}, \forall l \in \{1,...,L(K)\} \tag{5.43}$$

$$\sum_{p\in S_K^-(l)} w_i^p \leqslant 1 - v_i^l \qquad \forall i \in \{1,...,n\}, \forall l \in \{1,...,L(K)\} \tag{5.44}$$

$$v_i^l \in \{0,1\} \qquad \forall i \in \{1,...,n\}, \forall l \in \{1,...,L(K)\} \tag{5.45}$$

$$w_i^k \geqslant 0 \qquad \forall i \in \{1,...,n\}, \forall k \in \{1,...,K\} \tag{5.46}$$

$$x \in [0,1]^n \tag{5.47}$$

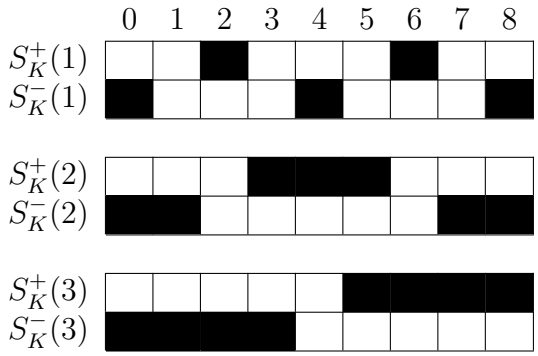
$$t \in \mathbb{R}^A \tag{5.48}$$

$$z_i \in [e^{-\bar{\gamma}_i}, 1] \qquad \forall i \in \{1,...,n\} \tag{5.49}$$

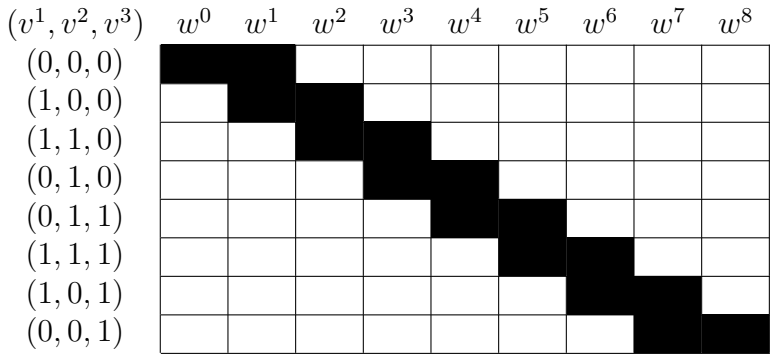
Where $L(K) = \lceil \log_2 K \rceil$ *and for any* $l \in \{1,...,L(K)\}$*:*

$$S_K^+(l) := \{p \in \{0,...,K\} : \forall q \in Q_K(p), (B_K(q))_l = 1\}$$
$$S_K^-(l) := \{p \in \{0,...,K\} : \forall q \in Q_K(p), (B_K(q))_l = 0\}$$

Where $Q_K(p) := \{q \in \{1,...,K\} : p \in \{q-1,q\}\}$ *and* $B_K : \{1,...,K\} \longmapsto \{0,1\}^{L(K)}$ *is a Gray code.*



(a) Example of sets $\left(\left\{S_K^+(l), S_K^-(l)\right\}\right)_{l\in\{1,..,\lceil \log_2 K\rceil\}}$



(b) All combinations of extra variables v^l and effect over variables w^k

Figure 5.2: Example of sets $\left(\left\{S_K^+(l), S_K^-(l)\right\}\right)_{l\in\{1,..,\lceil \log_2 K\rceil\}}$ and their implications for $K = 8$

We can solve this nonlinear mixed integer problem by Branch and Bound solving its successive continuous relaxations with interior point methods and use the branching scheme suggested in [79].

5.5 Solution quality

To compare a risk neutral solution with a risk averse solution, we want to see if there is some kind of stochastic dominance of a risk averse strategy versus a risk neutral one. To do so, we want to compare the payoffs distributions of the defender depending of its risk aversion. In a real situation, the defender covers m targets out of the n and the attacker targets a single place.

The only possible outcomes for the defender are: 1)being attacked by a type a adversary on a defended target i with payoff $V = \bar{R}_i^a > 0$ or 2)being attacked by a type a adversary on an undefended target i with payoff $V = \bar{P}_i^a < 0$. Consequently, if we assume that all the payoffs $\bar{R}_i^a$ and $\bar{P}_i^a$ are different the only values possible are in

$$V \in \{V_1 < V_2 < ... < V_{2An-1} < V_{2An}\} = \bigcup_{i=1}^{n}\bigcup_{a=1}^{A}\left\{\bar{R}_i^a, \bar{P}_i^a\right\}$$

Let see what is the probability to obtain each of these outcomes. Given a mixed defense strategy $x \in [0,1]^n$ and the associated quantal response, the probability to block an attack at target i is:

$$\mathbb{P}\left[V = \bar{R}_i^a\right] = p_a x_i \frac{e^{\lambda_a\left(x_i P_i^a + (1-x_i)R_i^a\right)}}{\sum_{j=1}^{n} e^{\lambda_a\left(x_j P_j^a + (1-x_j)R_j^a\right)}}$$

and the probability to undergo a type a attack at a defenseless target i is:

$$\mathbb{P}\left[V = \bar{P}_i^a\right] = p_a(1-x_i) \frac{e^{\lambda_a\left(x_i P_i^a + (1-x_i)R_i^a\right)}}{\sum_{j=1}^{n} e^{\lambda_a\left(x_j P_j^a + (1-x_j)R_j^a\right)}}$$

This way we can compute the probability distribution of the payoff of any defender with quantal response adversary without sampling a large number of simulations.

5.6 Conclusions

In this chapter, we extended the classic model of Stackelberg security games with quantal response (SSGQR). We showed a way to optimally solve in polynomial time a risk averse version of SSGQR. Furthermore we proposed a Branch and Bound scheme to solve an approximation of the problem when facing different types of adversaries. The models proposed, however, result in difficult optimization problems as there are several exponentials causing numerical issues. Our current work consists in tackling this numerical instability and improve the accuracy of our formulations.

Chapter 6

Conclusion

In this thesis, we first developed fast and accurate algorithms projecting GPS data on transportation networks. We designed a computationally efficient method able to dynamically match GPS trajectories in an online setting. We then defined a new error measure and a computationally fast algorithm to find the associated optimal matching with respect to this error measure that has the extra ability to detect and project cycles. These methods provide us the data needed to solve real instances of Risk Averse Shortest Path (RASP) problems. Further, they have the potential to allow fast and precise online localization of vehicles on transportation networks.

In Chapter 2, we then developed solution schemes for risk averse combinatorial optimization problems. We showed that problems minimizing an Entropic risk measure were difficult to solve with a good stochastic guarantee of the solution returned. The combinatorial nature of our base problem prohibits the use of extremely large samples. Combined with the potentially high sensitivity of the entropic risk measure to "bad" realizations - hence needing large samples to stabilize its behavior - it leads to a clear computational bottleneck. Although encouraging computational results were found for CVaR, the time needed to obtain stochastically good solutions for real instances still proved to be too long for emergency dispatching. In a future work we want to investigate the use of Importance Sampling techniques to reduce the computational burden of our problems without lowering the quality of the returned solutions. Also, having showed that the decomposition technique of [35] is applicable to CVaR minimization problem, we want to investigate further the structure of this method. More specifically, we want to know if we can quickly find a minimal partition of the scenarios such that the relaxation it induces leads to an optimal solution of the disaggregated original problem. Another future line of work we want to explore is the central role that CVaR has for coherent risk measures: [48] proved that any coherent risk measure is representable as a convex combination of CVaRs at different security levels. [69] presents a practical approach of this result. Using the decomposition methods for CVaR presented in this thesis, we want to derive efficient solution schemes for minimization problems with coherent risk measures objectives. We also want to apply directly the methodology described in this chapter to solve risk averse facility location problems as the routing from the facilities to the customers is an important part of them.

The preliminary work of Chapter 3 introduced novel formulations that can be theoretically useful to tackle difficult path problems in a faster way. We introduce the first formulation for the TSP having $O\left(n \log_2 \frac{m}{n}\right)$ zero-one variables. Furthermore, we introduce new extended formulations and branching schemes for *st*-path and *st*-cut problems in Directed Acyclic Graphs. In future work, we want to test these formulations to check if they provide a computational advantage against state of the art solvers like [27]. Further, we noticed that the extended formulation introduced by [79] increases the size of the formulation thus potentially slowing the LP solution and possibly destroying any special structure in the original formulation. On another hand, the extended formulation allows the use of all the machinery of typical solvers containing advanced routines for branch and node selection automatically. So it seems natural to compare the computational performances of the direct resolution of an extended formulation against a tailored LP resolution (e.g.: interior point methods using shortest path subroutines) combined with the constraint branching scheme induced by the extended formulation. In this chapter, we also showed that for particular network problems, we can cover the entire set of binary variables by disjunctive constraints and consequently reduce the binary variables number significantly. We also want to investigate if we can find such complete or - more likely - partial coverings for generic combinatorial problems.

The last chapter introduced a new type of Quantal Response Equilibrium in Security games (QRES) where the defender is risk averse. We show that we can solve it in polynomial time using a trick from [91], and we extend the model when there is several types of adversary (QRESMA). We could not solve this last type of problem in polynomial time but instead we formulated it as a MINLP and reduced its number of integer variables thanks to a result of [79]. The formulations obtained for the risk averse version of QRES include a number of exponential functions, which are likely to lead to problems that are particularly unstable numerically. Future work should take into account this instability using robust algorithms. For QRESMA, we also should consider Branch and Bound algorithms with tailored methods for solving the continuous relaxations and the use of specialized branching schemes ([79]).

Appendix

In this appendix, we show some extra computational results of the methods described in chapter 3.

Param	Value	$\mu_{S,T}$	$\sigma^2_{S,T}$	$L_{S,T}$	$U_{S'}$	gap[%]	t[s]
Σ	$\neq I$	9011.2	137.3	8955.7	9208.4	1.9	2975.5
	I	17291.4	924.1	16937.7	18206.5	4.2	92.9
c	0.5	8938.9	148.7	8881.7	9013.8	1.2	75.3
	2	17291.4	924.1	16937.7	18206.5	4.2	92.9
S	500	16187.8	1573.1	15586.5	19007.2	10.3	32.8
	1000	16864.8	1206.8	16403.2	18658.0	6.8	53.4
	2000	17291.4	924.1	16937.7	18206.5	4.2	92.9
	5000	17461.4	591.5	17234.7	18058.9	3.0	207.3
	10000	17532.9	440.2	17363.9	18007.8	2.4	395.7
T	2	17149.4	759.6	15699.1	18206.5	9.3	8.1
	10	17203.9	868.0	16462.5	18206.5	6.1	19.2
	20	17276.1	875.4	16747.1	18206.5	4.9	43.5
	30	17276.1	920.3	16821.8	18206.5	4.6	67.6
	40	17298.2	915.6	16906.6	18206.5	4.3	89.4
	50	17291.4	924.1	16937.7	18206.5	4.2	92.9
S'	40000	17291.4	924.1	16937.7	18721.3	5.9	92.9
	80000	17291.4	924.1	16937.7	18466.6	5.0	92.9
	120000	17291.4	924.1	16937.7	18318.8	4.5	92.9
	160000	17291.4	924.1	16937.7	18268.2	4.3	92.9
	200000	17291.4	924.1	16937.7	18206.5	4.2	92.9
ε	0.01	42809.3	3419.2	41501.6	47392.4	12.3	119.3
	0.05	27278.1	1374.0	26751.9	28288.1	5.4	128.8
	0.1	21161.1	875.2	20826.0	21714.1	4.1	121.7
	0.5	10083.6	284.6	9974.3	10173.0	1.9	83.3
	0.9	6843.2	179.4	6774.3	6901.7	1.8	66.5
	0.95	6547.6	171.1	6481.8	6605.4	1.9	65.6
	0.99	6316.7	165.1	6253.6	6370.7	1.8	65.4

Table 1: CVaR framework Vs. parameters

Param	Value	$\mu_{S,T}$	$\sigma^2_{S,T}$	$L_{S,T}$	$U_{S'}$	gap[%]	t[s]
Σ	$\neq I$	6515.1	70.7	6487.7	6560.9	1.1	693.8
	I	29142.3	5434.4	27065.9	105480.5	57.3	151.1
c	0.5	7585.1	251.3	7308.0	10914.5	19.3	82.5
	2	28267.1	4619.9	23174.6	105480.5	60.8	151.1
S	500	20307.3	4324.7	18655.2	135362.5	71.2	88.6
	1000	24253.1	4981.3	22349.9	125367.5	65.9	101.7
	2000	29142.3	5434.4	27065.9	105480.5	57.3	151.1
	5000	35060.5	4594.4	33304.0	104970.2	50.7	340.3
	10000	38767.0	4976.9	36864.7	109026.7	52.8	540.7
T	2	28088.4	3129.0	22114.6	105480.5	62.3	8.2
	10	28159.8	4629.6	24206.7	105480.5	59.8	25.1
	20	28944.4	4861.1	26008.9	105480.5	58.3	70.3
	30	29131.0	5418.0	26459.3	105480.5	57.9	102.6
	40	29108.5	5328.0	26832.8	105480.5	57.5	110.3
	50	29142.3	5434.4	27065.9	105480.5	57.3	151.1
S'	40000	29142.3	5434.4	27065.9	81573.9	52.7	151.1
	80000	29142.3	5434.4	27065.9	91536.6	55.3	151.1
	120000	29142.3	5434.4	27065.9	98024.9	56.8	151.1
	160000	29142.3	5434.4	27065.9	97793.1	56.3	151.1
	200000	29142.3	5434.4	27065.9	105480.5	57.3	151.1
α	1000	53695.5	9564.2	50040.1	185968.0	72.1	149.0
	1500	50291.9	9481.9	46669.9	180529.0	73.1	161.3
	2000	46996.3	9360.3	43419.8	173733.2	74.0	173.8
	5000	29412.1	7952.3	26374.4	146096.3	80.7	288.8
	20000	8524.6	768.6	8230.8	27925.2	54.8	109.3
	25000	7915.2	565.4	7699.1	16216.3	36.3	94.6
	40000	7160.3	348.5	7027.1	7895.3	10.3	80.6

Table 2: Entropy framework Vs. parameters

Bibliography

[1] S. Agrawal, Y. Ding, A. Saberi, and Y. Ye. Correlation robust stochastic optimization. In *Proceedings of the 21st annual ACM-SIAM Symposium on discrete algorithms, Austin, TX*, pages 1087–1096, 2010.

[2] S. Agrawal, Y. Ding, A. Saberi, and Y. Ye. Price of correlations in stochastic optimization. *Operations Research*, 60(1):150–162, 2012.

[3] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network flows: theory, algorithms, and applications*. Prentice hall, 1993.

[4] J. A. Appleget and R. K. Wood. *Explicit-constraint branching for solving mixed-integer programs*. Springer, 2000.

[5] K. J. Arrow. *Aspects of the theory of risk-bearing*. Yrjö Jahnssonin Säätiö, 1965.

[6] P. Artzner, F. Delbaen, J.-M. Eber, and D. Heath. Thinking coherently: Generalized scenarios rather than VaR should be used when calculating regulatory capital. *RISK-LONDON-RISK MAGAZINE LIMITED-*, 10:68–71, 1997.

[7] P. Artzner, F. Delbaen, J.-M. Eber, and D. Heath. Coherent measures of risk. *Mathematical finance*, 9(3):203–228, 1999.

[8] A. Atamtürk and M. Zhang. Two-stage robust network flow and design under demand uncertainty. *Operations Research*, 55(4):662–673, 2007.

[9] I. Averbakh. On the complexity of a class of combinatorial optimization problems with uncertainty. *Mathematical Programming*, 90(2):263–272, 2001.

[10] M. Avriel and A.C. Williams. The value of information and stochastic programming. *Operations Research*, 18(5):947–954, 1970.

[11] E. Balas. Disjunctive programming. *Annals of Discrete Mathematics*, 5:3–51, 1979.

[12] E. Balas. Disjunctive programming and a hierarchy of relaxations for discrete optimization problems. *SIAM Journal on Algebraic Discrete Methods*, 6(3):466–486, 1985.

[13] E. M. L. Beale and J. A. Tomlin. Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables. *OR*, 69(447–

454):99, 1970.

[14] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust optimization*. Princeton University Press, 2009.

[15] D. Bertsimas and M. Sim. The price of robustness. *Operations research*, 52(1):35–53, 2004.

[16] J. R. Birge and F. Louveaux. *Introduction to stochastic programming*. Springer Science & Business Media, 2011.

[17] C.A. Blazquez. A decision-rule topological map-matching algorithm with multiple spatial data. In *Global Navigation Satellite Systems: Signal, Theory and Applications*, pages 215–240. InTech, 2012.

[18] C.F. Camerer, T.-H. Ho, and J.-K. Chong. A cognitive hierarchy model of games. *The Quarterly Journal of Economics*, 119(3):861–898, 2004.

[19] F. Chen, M. Shen, and Y. Tang. Local path searching based map matching algorithm for floating car data. *Procedia Environmental Sciences*, 10:576–582, 2011.

[20] R. Chicoisne. Solving hard integer problems on networks: Branching strategies. *Working paper*, 2015.

[21] R. Chicoisne, D. Espinoza, M. Goycoolea, E. Moreno, and E. Rubio. A new algorithm for the open-pit mine production scheduling problem. *Operations research*, 60(3):517–528, 2012.

[22] R. Chicoisne, F. Ordóñez, and D. Espinoza. Efficient algorithms to match gps data on a map. *Submitted to Annals of Operations Research*, 2014.

[23] R. Chicoisne, F. Ordóñez, and D. Espinoza. Risk averse shortest paths: A computational study. *To be submitted to Mathematical Programming*, 2015.

[24] R. Chicoisne, F. Ordóñez, and M. Tambe. Optimal risk averse defense strategies against quantal response in security games. *Working paper*, 2015.

[25] E. Ciurea and L. Ciupalâ. Sequential and parallel algorithms for minimum flows. *Journal of Applied Mathematics and Computing*, 15(1-2):53–75, 2004.

[26] C.E. Cortés, J. Gibson, A. Gschwender, M. Munizaga, and M. Zúñiga. Commercial bus speed diagnosis based on gps-monitored data. *Transportation Research Part C: Emerging Technologies*, 19(4):695–707, 2011.

[27] IBM ILOG CPLEX. *V12. 1: Users Manual for CPLEX*, 2009.

[28] I.R. De Farias, E.L. Johnson, and G.L. Nemhauser. Branch-and-cut for combinatorial optimization problems without auxiliary binary variables. *The Knowledge Engineering Review*, 16(01):25–39, 2001.

[29] I.R. de Farias Jr, E.L. Johnson, and G.L. Nemhauser. A generalized assignment problem with special ordered sets: a polyhedral approach. *Mathematical Programming*, 89(1):187–203, 2000.

[30] G. Di Battista, P. Eades, R. Tamassia, and I.G. Tollis. *Graph drawing: algorithms for the visualization of graphs*. Prentice Hall PTR, 1998.

[31] D.H. Douglas and T.K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973.

[32] A. Drexl and A. Kimms. Lot sizing and schedulingsurvey and extensions. *European Journal of Operational Research*, 99(2):221–235, 1997.

[33] E. Drezner. Facility location: A survey of applications and methods. *Journal of the Operational Research Society*, 47(11):1421–1421, 1996.

[34] A. Echeverría, R. Chicoisne, D. Espinoza, and F. Ordóñez. Redesign of the dispatching system for the santiago fire department. *Working paper*, 2015.

[35] D. Espinoza and E. Moreno. A primal-dual aggregation algorithm for minimizing conditional value-at-risk in linear programs. *Computational Optimization and Applications*, 59(3):617–638, 2014.

[36] B.A. Foster and D.M. Ryan. An integer programming approach to the vehicle scheduling problem. *Journal of the Operational Research Society*, 27(2):367–384, 1976.

[37] G. Ghiani, F. Guerriero, G. Laporte, and R. Musmanno. Real-time vehicle routing: Solution concepts, algorithms and parallel computing strategies. *European Journal of Operational Research*, 151(1):1–11, 2003.

[38] E.N. Gilbert. Gray codes and paths on the n-cube. *Bell System Technical Journal*, 37(3):815–826, 1958.

[39] P.A. Haile, A. Hortaçsu, and G. Kosenok. On the empirical content of quantal response equilibrium. *The American Economic Review*, 98(1):180–200, 2008.

[40] H.W. Hamacher and Z. Drezner. *Facility location: applications and theory*. Springer, 2002.

[41] R. Jans and Z. Degraeve. Modeling industrial lot sizing problems: a review. *International Journal of Production Research*, 46(6):1619–1643, 2008.

[42] A.B. Keha, I.R. de Farias Jr, and G.L. Nemhauser. A branch-and-cut algorithm without binary variables for nonconvex piecewise linear optimization. *Operations research*, 54(5):847–858, 2006.

[43] C. Kiekintveld, M. Jain, J. Tsai, J. Pita, F. Ordóñez, and M. Tambe. Computing optimal randomized resource allocations for massive security games. In *Proceedings of*

the 8th AAMAS Conference, Budapest, Hungary, volume 1, pages 689–696. International Foundation for AAMAS, 2009.

[44] A.J. Kleywegt, A. Shapiro, and T. Homem-de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502, 2002.

[45] D. E. Knuth. *The art of computer programming: sorting and searching*, volume 3. Pearson Education, 1998.

[46] P. Kouvelis and G. Yu. *Robust discrete optimization and its applications*, volume 14. Springer, 1997.

[47] A. Künzi-Bay and J. Mayer. Computational aspects of minimizing conditional value-at-risk. *Computational Management Science*, 3(1):3–27, 2006.

[48] S. Kusuoka. On law invariant coherent risk measures. *Advances in mathematical economics*, 3(1):83–95, 2001.

[49] K. Lakakis, P. Savvaidis, I. M. Ifadis, and D.I. Doukas. Quality of map matching procedures based on DGPS and stand alone GPS positioning in an urban area. In *Proceedings of FIG Working Week, Athens, Greece*, pages 22–27, 2004.

[50] G. Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(3):345–358, 1992.

[51] J. Lee. All-different polytopes. *Journal of Combinatorial Optimization*, 6(3):335–352, 2002.

[52] J. Lee and F. Margot. On a binary-encoded ILP coloring formulation. *INFORMS Journal on Computing*, 19(3):406–415, 2007.

[53] H. Levy. *Stochastic dominance: Investment decision making under uncertainty*, volume 12. Springer Science & Business Media, 2006.

[54] A. E.-B. Lim, J.G. Shanthikumar, and G.-Y. Vahn. Conditional value-at-risk in portfolio optimization: Coherent but fragile. *Operations Research Letters*, 39(3):163–171, 2011.

[55] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang. Map-matching for low-sampling-rate gps trajectories. In *Proceedings of the 17th ACM SIGSPATIAL, Seattle, WA*, pages 352–361. ACM, 2009.

[56] F. Marchal, J. Hackney, and K. W. Axhausen. Efficient map matching of large global positioning system data sets: Tests on speed-monitoring experiment in zürich. *Transportation Research Record: Journal of the Transportation Research Board*, 1935:93–100, 2005.

[57] H. Markowitz. Portfolio selection. *The journal of finance*, 7(1):77–91, 1952.

[58] R.D. McKelvey and T.R. Palfrey. Quantal response equilibria for normal form games. *Games and economic behavior*, 10(1):6–38, 1995.

[59] A. Mehrotra and M.A. Trick. A branch-and-price approach for graph multi-coloring. In *Extending the horizons: Advances in computing, optimization, and decision technologies*, pages 15–29. Springer, 2007.

[60] R. B. Myerson. *Game theory*. Harvard university press, 2013.

[61] P. Newson and J. Krumm. Hidden markov map matching through noise and sparseness. In *Proceedings of the 17th ACM SIGSPATIAL, Seattle, WA*, pages 336–343. ACM, 2009.

[62] E. Nikolova, M. Brand, and D.R. Karger. Optimal route planning under uncertainty. In *Proceedings of International Conference on Automated Planning and Scheduling, Ambleside, U.K.*, 2006.

[63] J. Pita, M. Jain, J. Marecki, F. Ordóñez, C. Portway, M. Tambe, C. Western, P. Paruchuri, and S. Kraus. Deployed armor protection: the application of a game theoretic model for security at the los angeles international airport. In *Proceedings of the 7th AAMAS: industrial track, Cascais Miragem, Portugal*, pages 125–132. International Foundation for AAMAS, 2008.

[64] J. Pita, M. Jain, F. Ordóñez, M. Tambe, and S. Kraus. Solving stackelberg games in the real-world: Addressing bounded rationality and limited observations in human preference models. *Artificial Intelligence Journal*, 174(15):1142–1171, 2010.

[65] J. W. Pratt. Risk aversion in the small and in the large. *Econometrica: Journal of the Econometric Society*, 32(1/2):122–136, 1964.

[66] M. A. Quddus, R. B. Noland, and W. Y. Ochieng. A high accuracy fuzzy logic based map matching algorithm for road transport. *Journal of Intelligent Transportation Systems*, 10(3):103–115, 2006.

[67] R.T. Rockafellar and S. Uryasev. Optimization of conditional value-at-risk. *Journal of risk*, 2:21–42, 2000.

[68] D.M. Ryan and B. A. Foster. An integer programming approach to scheduling. *Computer scheduling of public transport urban passenger vehicle and crew scheduling*, pages 269–280, 1981.

[69] A. Shapiro. On kusuoka representation of law invariant risk measures. *Mathematics of Operations Research*, 38(1):142–152, 2013.

[70] A. Shapiro, D. Dentcheva, and A. Ruszczyński. *Lectures on stochastic programming: modeling and theory*, volume 16. SIAM, 2014.

[71] L.V. Snyder. Facility location under uncertainty: a review. *IIE Transactions*, 38(7):547–564, 2006.

[72] D.O. Stahl II and P.W. Wilson. Experimental evidence on players' models of other players. *Journal of economic behavior & organization*, 25(3):309–327, 1994.

[73] G. Tintner. Stochastic linear programming with applications to agricultural economics. In *Proceedings of the Second Symposium in Linear Programming, Washington, DC*, volume 1, pages 197–228. National Bureau of Standards Washington, D. C, 1955.

[74] S. Vajda. *Mathematical Programming*. Addison Wesley,London, 1961.

[75] F. Vanderbeck and L.A. Wolsey. An exact algorithm for IP column generation. *Operations Research Letters*, 19(4):151–159, 1996.

[76] François Vanderbeck. Implementing mixed integer column generation. In G. Desaulniers, J. Desrosiers, and M.M. Solomon, editors, *Column Generation*, pages 331–358. Springer, 2005.

[77] François Vanderbeck. Branching in branch-and-price: a generic scheme. *Mathematical Programming*, 130(2):249–294, 2011.

[78] J.P. Vielma. Mixed integer linear programming formulation techniques. *SIAM Review*, 57:3–57, 2015.

[79] J.P. Vielma and G.L. Nemhauser. Modeling disjunctive constraints with a logarithmic number of binary variables and constraints. *Mathematical Programming*, 128(1-2):49–72, 2011.

[80] J. Von Neumann and O. Morgenstern. Theory of games and economic behavior. *Princeton University Press, Princeton*, 1944.

[81] H. Von Stackelberg. *The theory of the market economy*. William Hodge, 1952.

[82] B. S. Westgate, D. B. Woodward, D.S. Matteson, and S. G. Henderson. Large-network travel time distribution estimation, with application to ambulance fleet management. *Under review*, 2013.

[83] R. J.-B. Wets. Programming under uncertainty: the equivalent convex program. *SIAM Journal on Applied Mathematics*, 14(1):89–105, 1966.

[84] C.E. White, D. Bernstein, and A.L. Kornhauser. Some map matching algorithms for personal navigation assistants. *Transportation Research Part C: Emerging Technologies*, 8(1):91–108, 2000.

[85] J.R. Wright and K. Leyton-Brown. Beyond equilibrium: Predicting human behavior in normal-form games. In *Proceedings of the 24th AAAI conference on artificial intelligence, Atlanta, GA*, 2010.

[86] M. E. Yaari. The dual theory of choice under risk. *Econometrica*, 55(1):95–115, 1987.

[87] M.E. Yaari. Some remarks on measures of risk aversion and on their uses. *Journal of*

Economic theory, 1(3):315–329, 1969.

[88] B. Yang, C. Guo, and C.S. Jensen. Travel cost inference from sparse, spatio temporally correlated time series using markov models. In *Proceedings of the VLDB Endowment, Riva del Garda, Italy*, volume 6, pages 769–780. VLDB Endowment, 2013.

[89] J. Yang, S. Kang, and K. Chon. The map matching algorithm of gps data with relatively long polling time intervals. *Journal of the Eastern Asia Society for Transportation Studies*, 6:2561–2573, 2005.

[90] R. Yang, C. Kiekintveld, F. Ordóñez, M. Tambe, and R. John. Improving resource allocation strategy against human adversaries in security games. In *22th IJCAI Proceedings, Barcelona, Spain*, volume 22, pages 458–464. AAAI Press, 2011.

[91] R. Yang, F. Ordóñez, and M. Tambe. Computing optimal strategy against quantal response in security games. In *Proceedings of the 11th AAMAS, Valencia, Spain*, volume 2, pages 847–854. International Foundation for AAMAS, 2012.

[92] L. Zhang and T. Homem-de Mello. An optimal path model for the risk-averse traveler. Technical report, Working paper, School of Business, Universidad Adolfo Ibañez, Santiago, Chile, 2014.

Vita

Renaud Chicoisne was born in Gien, France, on 21 December 1984. After two internships in Universidad Adolfo Ibañez in Santiago, Chile, he received in 2009 his Mathematical Engineering Title from CUST and his Master of Science in Operations Research from the Université Blaise Pascal, both in Clermont-Ferrand, France. After working as a research engineer at the Université de Bordeaux-1 in Talence, France, he enrolled in the Department of Industrial Engineering at the Universidad de Chile in Santiago, Chile in March of 2010 where he completed his doctoral research on efficient algorithms for risk averse optimization problems.