Robust Planning and Scheduling using Column Generation

PhD Thesis

Andrew Murray

Department of Computer and Information Sciences

University of Strathclyde, Glasgow

June 4, 2024

Abstract

Autonomous Planning and Scheduling (APS) is the problem of autonomously planning, scheduling and executing a sequence of actions in an environment to achieve some goals. Due to the increased adoption of APS in sensitive applications, there is an increasing need for models that are robust to the uncertainty prevalent in the real world. Robustness can be achieved by modelling the APS problem via stochastic optimization, however such problems are inherently difficult to solve. Recently, the column generation method has shown precedent in solving stochastic optimization problems. In this technique, the main optimization problem is decomposed into two more manageable sub problems which are then solved iteratively: the Restricted Master Problem (RMP) and the Column Generation Problem (CGP). In this thesis, we apply column generation in a novel way to develop robust solutions to APS problems.

The first problem we address is a 5G telecommunications planning problem called the Virtual Network Function Placement and Routing Problem (VNF-PRP). In 5G, Internet Service Providers (ISP) must deliver tailored network services for a variety of use cases; often with heterogeneous requirements defining the expected Quality of Service (QoS). Past approaches at solving this problem do not consider all of the constraints required to guarantee QoS. Likewise, the majority of prior algorithms either solve a large Integer Linear Program (ILP), which is computationally intractable for practical problems; or leverage heuristics, which give no guarantees on solution quality. In this thesis, we present a column generation based VNF-PRP algorithm which solves: the RMP, which optimizes the placement, replication and routing of the VNFs given the paths generated so far; and the CGP, which generates new paths. Our approach is the first VNF-PRP algorithm to consider throughput, latency and availability constraints. It is also the first that is capable of computing a valid VNF placement, routing and replication solution, while providing a measure of solution quality. We validate our approach on a realistic Mobile Edge Cloud (MEC) network architecture and show that our model can find near optimal solutions to practical sized problems within a reasonable time.

The second problem we address is a scheduling problem called Strong Controllability (SC). SC of Probabilistic Simple Temporal Networks (PSTN) involves finding a schedule to execute a sequence of actions that maximises the probability that all constraints are satisfied (robustness). Previous approaches to this problem assume independence of probabilistic durations. This gives no guarantee of finding the schedule optimising robustness, and fails to consider correlations between action durations that frequently arise in practical applications. In this thesis, we formally define the Correlated Simple Temporal Network (Corr-STN), which generalises the PSTN by removing the restriction of independence, and show that the problem of Corr-STN SC is convex. We present the first Corr-STN SC algorithm based on column generation which solves: the RMP, which finds the most robust schedule given an approximation of the joint distribution; and the CGP, which finds a new point to refine the approximation. We validate our approach on a number of Corr-STNs and find that our method offers strictly more robust solutions when compared with prior PSTN SC approaches.

Publications

1. "Constraint Relaxation Costs in Expected Value Probabilistic Simple Temporal Networks", Andrew Murray, International Conference on Automated Planning and Scheduling (Doctoral Consortium), 2021, (Reviewed, but not formally published)

2. "Towards Temporally Uncertain Explainable AI Planning", Andrew Murray, Benjamin Krarup and Michael Cashmore, International Conference on Distributed Computing and Internet Technology, 2022

3. "Joint Chance Constrained Probabilistic Temporal Networks via Column Generation (Extended Abstract)", Andrew Murray, Michael Cashmore, Ashwin Arulselvan and Jeremy Frank, International Symposium on Combinatorial Search, 2022 (Chapter 5)

4. "A Column Generation Approach to Correlated Simple Temporal Networks", Andrew Murray, Ashwin Arulselvan, Michael Cashmore, Marc Roper and Jeremy Frank, International Conference on Automated Planning and Scheduling, 2023 (Chapter 5)

5. "The Cost of Quality of Service: SLA Aware VNF Placement and Routing using Column Generation", Andrew Murray, Ashwin Arulselvan, Marc Roper, Michael Cashmore, Swarup Mohalik, Ian Burdick and Sushanth David, International Workshop on Resilient Networks Design and Modelling, 2023 (Chapter 4)

Contents

List of Figures

List of Tables

Source Code

The source code used in this thesis is available online at the following repositories:

- Chapter 4: https://anonymous.4open.science/r/VNFPP-CG-00DC
- Chapter 5: https://anonymous.4open.science/r/CORRSTN-3E78

Acronyms

5G	5th Generation.
AI	Artificial Intelligence.
AIA	Adaptive Interference Aware.
APS	Automated Planning and Scheduling.
BFS	Basic Feasible Solution.
CC-PSTN	Chance Constrained Probabilistic Simple Temporal Network.
CDF	Cumulative Density Function.
CGP	Column Generation Problem.
Corr-STN	Correlated Simple Temporal Network.
DC	Data Center.
eMBB	Enhanced Mobile Broadband.
ILP	Integer Linear Program.
ISP	Internet Service Provider.
KPI	Key Performance Indicator.

LP Linear Program.

LRMP Linear Restricted Master Problem.

MDP Markov Decision Process.

MEC Mobile Edge Cloud.

MIP Mixed Integer Program.

MIQCP Mixed Integer Quadratically Constrained Program.

mMTC Massive Machine Type Communications.

MP Master Problem.

MTBF Mean Time Between Failures.

MTTR Mean Time To Repair.

NF Network Function.

NFV Network Function Virtualization.

NLP Nonlinear Program.

OR Operations Research.

PDDL Planning Domain Definition Language.

PSTN Probabilistic Simple Temporal Network.

QoS Quality of Service.

RMP Restricted Master Problem.

SC Strong Controllability.

SFC	Service Function Chain.
SLA	Service Level Agreement.
STN	Simple Temporal Network.
STNU	Simple Temporal Network with Uncertainty.
UHD	Ultra High Definition.
URLLC	Ultra Reliable Low Latency Communications.
VNF	Virtual Network Function.
VNF-FG	Virtual Network Function Forwarding Graph.
VNF-PP	Virtual Network Function Placement Problem.
VNF-PRP	Virtual Network Function Placement and Routing Problem.

Preface/Acknowledgements

Most importantly, I would like to thank my academic supervisors for all of their guidance and encouragement throughout the course of my PhD. I was fortunate enough to have three excellent supervisors whose expertise was invaluable. I'd like to thank Michael Cashmore for all the assistance with planning, scheduling and more generally programming; as well as for providing excellent developmental support in my initial few years of study. I'd also like to thank Ashwin Arulselvan for his mathematical expertise and for taking the time to explain technical ideas in an intuitive way. Finally, I'd like to thank Marc Roper for taking up the reins in my final year. Regardless of the time or day, he was available to answer my questions and provide crucial guidance on the development of this thesis.

I was also very fortunate to collaborate with a number of external partners whose guidance made this thesis possible. I'd like to thank the AI research team at Ericsson for introducing me to many interesting problems in 5G, as well as for welcoming me so kindly to their office during my research visit to Stockholm. In particular, I would like to thank Swarup Mohalik whose knowledge of telecommunications and innovative ideas helped make Chapter 4 possible. In addition, I would like to thank Jeremy Frank for his guidance and insights into controllability which helped sculpt Chapter 5 of this thesis. This work would also not have been possible without all the constructive feedback provided by anonymous reviewers for publications related to this thesis.

Outside of the work presented in this thesis, a number of people were fundamental in my professional development as a researcher. The staff in CIS provided me the opportunity to work as a lab demonstrator in their classes. Working as a lab demonstrator gave me a welcome break from research and introduced me to a host of new techniques and ideas. I'd also like to thank the staff at JP Morgan AI research for their guidance during my 6 month internship and for highlighting the applicability of all the skills and experiences gained throughout my PhD.

This work was supported by a number of grants. The Engineering and Physical Sciences Research Council studentship helped make this thesis financially possible. Similarly, the Scottish Informatics and Computer Science Alliance Saltire emerging researcher grant helped fund my visit to Ericsson.

Finally, I would like to thank all of my family and friends for helping take my mind off research. This thesis would have been significantly more challenging if it weren't for the many hiking and climbing trips that kept me entertained throughout these past years. My girlfriend Ailsa, for all her love, support and the countless morning coffees; and my parents, for supporting my decision to pursue a PhD.

Chapter 1

Introduction

1.1 Background

Planning and scheduling are two interconnected problems which are at the root of many decision processes. A planning problem involves selecting a sequence of actions, and ordering them so as to achieve some goals. For example, assume that I was at my house and was tasked with delivering some groceries to a friend. In order to achieve this, I may have to walk to the supermarket, pick up the groceries, walk to my friends house and finally, deliver the groceries. This sequence of actions can be considered a plan which achieves the goal condition: my friend has the groceries. On the other hand, scheduling assumes that the actions required to achieve the goals are known in advance, and deals with the problem of finding the best time to execute them. I may need to reason over possible travel times to decide the best time to leave my house. The assignment of times to actions is known as a schedule.

Because planning and scheduling problems are so prevalent, there has been a targeted effort to develop autonomous systems capable of automatically solving them. These autonomous systems are collectively known as automated planners and schedulers, while the field associated with developing them is known as Automated Planning and Scheduling (APS) or alternatively Artificial Intelligence (AI) Planning and Scheduling.

In many applications, planning and scheduling algorithms have matched or even surpassed human capabilities. A Monte Carlo tree search (a type of planning algorithm) based planning system was at the core of the AlphaGo algorithm, a computer program which managed to beat the worlds best player at the game of Go [3]. The game of Go features some 2×10^{170} possible states, far more than any human brain can deliberate over [4]. Despite this, developing APS models to work in real world applications, as opposed to games which often have well defined rules and logic, poses some additional challenges. One of the key challenges is: how to make the solution *robust*. This is becoming increasingly important as new high risk applications of AI [5–7] have led to a growing call for governmental regulation [8]. Recently, robustness has been identified by the European Union high level expert group on AI as one of the key pillars that new AI systems should be evaluated against in their "ethical guidelines for trustworthy AI" [9].

Robustness of a computer program refers to how well it can perform its functionality in the presence of perturbations, invalid inputs and stressful environmental conditions [10, 11]. In planning, Fox et al. define robustness as a measure of the likelihood that a plan will be executed successfully in the presence of uncertainty in the execution environment [12]. A similar definition has been provided in the context of scheduling by Fang et al. [13].

It's worth mentioning that the term robustness is often used within Operations Research (OR) in the context of *robust optimization* [14]. In robust optimization, the goal is to find the optimal solution that satisfies all uncertain values in a predefined uncertainty set. We stress here that, when we refer to robustness in this thesis we are referring to the definition implied in the APS literature, as opposed to robust optimization which is not covered. In this thesis, we address robust APS from a stochastic optimization perspective. Explicitly modelling the uncertainty via probabilities allows us to reason over the likelihood of the uncertainty sources, as well as providing quantifiable guarantees on robustness. However, doing so requires solving complicated optimization problems with a number of challenging characteristics. Dealing with probability distributions directly in the optimization can result in non-linear functions which render the optimization particularly difficult to solve.

Decomposition methods are a suite of techniques from OR which decompose difficult optimization problems into a number of smaller, more manageable sub problems. The column generation method, is one such technique which decomposes the problem into two problems which are then solved iteratively: a Restricted Master Problem (RMP) and a Column Generation Problem (CGP). The RMP solves a smaller optimization problem using a subset of the decision variables, while the CGP finds the best new variables (with an associated column of coefficients) to include in the next iteration of the RMP. The method terminates when no *improving* columns can be found: i.e. there are no new variables which, when included in the RMP, will improve the objective function value.

The column generation method was first introduced by Dantzig and Wolfe in their seminal paper, where they suggested iteratively adding columns to a Linear Program (LP) as required [15]. Since its conception, the column generation method has made it possible to solve efficiently many interesting classes of optimization problems which were previously intractable. Gilmore and Gomory were the first to apply the technique to a practical problem, by using it as a heuristic to solve the cutting stock problem [16, 17]. Desrosiers et al. embedded it within a branch and bound framework and used it to find solutions to vehicle routing problems with time windows, thus pioneering its use for large integer programs [18].

In this thesis we apply the column generation method in two new ways, to develop robust solutions to APS problems. In Chapter 4, we look at robustness in planning, in particular we study an application of combined task and path planning with robustness constraints known as the Virtual Network Function Placement and Routing Problem (VNF-PRP). To make the solution robust to failures, we introduce redundancies in the routing solution. In this problem, we use column generation to generate a set of feasible paths, thus decomposing the path planning from the robustness constraints. In Chapter 5, we look at robustness in scheduling, in particular we aim to develop schedules which are robust to correlated uncertainty. By encoding the problem using stochastic optimization we are able to explicitly optimize the robustness. Here, we use the column generation method to generate an approximation of the multi-variate distribution, which is then refined in the CGP.

1.2 Outline

We begin in Chapter 2 with a survey on robustness in APS. We introduce basic definitions of planning, scheduling and optimization and review literature related to achieving robustness in planning and scheduling, as well as techniques for optimization under uncertainty.

Since this thesis is intended as an application of the column generation method to new problems, we assume no prior knowledge of the underlying theory behind how it works. As a result, in Chapter 3 we provide an intuitive summary of the underlying principles and theory of the technique. We then demonstrate how this theory can be applied with reference to a classical application of the column generation method: the cutting stock problem.

In Chapter 4, we address a well known telecommunications planning problem, the Virtual Network Function Placement and Routing Problem (VNF-PRP). In the Network Function Virtualization (NFV) paradigm, Internet Service Providers (ISP) provide network services to customers by routing and processing traffic through an ordered sequence of Virtual Network Functions (VNF), for example a load balancer or firewall. In the 5th generation (5G) of mobile networks, multiple service use cases are hosted on a shared physical infrastructure in a process known as network slicing. The Quality of the Service (QoS) depends on the quantity and relative placement of the VNFs, and is quantified by a set of Key Performance Indicators (KPI) in a Service Level Agreement (SLA): a contract reached between the ISP and customer. This can be considered a planning problem since we are required to compute a sequence of actions (which VNFs to place on which nodes and which paths to configure for each service request) such that we achieve the goal of providing the service in line with the requirements outlined in the SLA. Because we do not know the routing paths in advance, the problem requires planning paths as well as tasks which renders classical planning techniques impractical. As a result, the problem is better modelled as a combinatorial optimization problem, which are typically solved in OR using branch and bound [19]. However, since there can be exponentially many paths on the network, and consequently exponentially many variables, branch and bound fails to find solutions except for the smallest instances. Instead, we use column generation as a heuristic and consider the valid routing *paths* as columns, such that we consider only the paths which are necessary. The column generation decomposition involves solving: the RMP which finds the placement and routing solution of the VNFs maximizing QoS (in terms of latency, throughput and availability), given the paths generated so far; and the CGP which generates new, improving paths.

In Chapter 5, we introduce and solve for the first time a scheduling problem known as Correlated Simple Temporal Network (Corr-STN) Strong Controllability (SC). This is an extension to the existing problem of Probabilistic Simple Temporal Network (PSTN) SC. PSTNs represent scheduling problems under temporal uncertainty. SC of PSTNs involves finding a schedule to a PSTN that maximises the probability that all constraints are satisfied (robustness). In Corr-STN SC, action durations are subject to *correlated* temporal uncertainty. We show that this is a convex optimization problem for multivariate Gaussian distributions meaning that the feasible region is convex. By taking a convex combination of approximation points denoted an *inner-approximation*, on the surface of the convex set, we can find the optimal solution. However, this can require exponentially many approximation points and therefore finding a solution means enumerating the non-linear function exponentially many times. Instead, we use column generation and consider *approximation points* as columns, such that we can consider only the approximation points required. The column generation procedure involves solving: the RMP which finds the most robust schedule using the approximation generated so far; and the CGP which computes the best new approximation point to include.

1.3 Motivation

Despite having its origins in the 1960s, the column generation method is typically not well understood or acknowledged outwith the realm of OR. This could be attributed to the fact that the majority of column generation literature approaches the subject from a highly theoretical perspective which is not so easily interpreted by researchers in other disciplines. Likewise, most applications and experimental studies focus on well defined OR problems containing very specific structures and characteristics. At the time of writing, a survey of the top 100 papers from a scholar search using the criteria “column generation” resulted in 84 papers published in OR specific conferences and journals and only 14 published in other fields. Likewise, of the 92 papers which contained some form of numerical study, 35 included vehicle routing problems, 18 included crew scheduling and 12 focused on the cutting stock problem. Applying column generation to new problems is not especially trivial; it requires technical expertise and knowledge about problem structure so that it can be exploited effectively. In a contrast to the norm, this thesis applies column generation to new types of problems. Through use of numerous practical examples, it is hoped that we can encourage further application of the technique to other problems.

In both planning and scheduling problems, there are often many potentially viable solutions. Selecting the best solution amongst a suite of viable solutions, falls within the domain of optimization and therefore both problems can be seen more generally as optimization problems. Despite this, the field of APS and OR have developed somewhat independently, with little dialogue between the two. This is primarily driven by a divergence in research focus: APS researchers tend to focus on the development of methods which are domain independent; that is they are applicable regardless of the problem they are applied to. In contrast, OR researchers tend to focus on developing techniques which excel at solving particular problems very well, but may not be applicable to other use cases. As a result, many techniques which have been developed for optimization problems in OR, are not utilised to their full extent by APS researchers and vice versa. Encouraging dialogue between these two fields was a further motivator for this thesis.

In Chapter 4, the problem we tackle can be considered a combined task and path planning problem, a particularly challenging class of planning problems. These types of problems are challenging in that we must reason over both the actions: where to assign the VNFs; and the paths: how do we route the service requests. The motivation for selecting this problem, is that the solution must be exceptionally robust to failures. The scope of services provided in 5G and beyond networks is vast, but expected use cases include remote surgery [20], automation of industrial machinery [21] and control of self driving cars [22]. The term availability refers to the proportion of time that a service is *available*: up-time versus the total time including downtime. The 3rd Generation Partnership Project (3GPP), the governing body in charge of maintaining industrial standards in mobile telecommunications networks, refers to high availability services as being available 99.999% of the time [23]. While this problem has been tackled in the past, prior approaches typically solve an exact Mixed Integer Program (MIP) which is not scalable, or use meta-heuristics which give no guarantee of solution quality. By drawing analogies to well-known OR problems, in particular vehicle routing problems, it became apparent that column generation was a good fit [24]. Using column generation allows us to solve the routing problem for each service independently as a shortest path problem which is computationally tractable. While column generation has been applied to this problem in the past, prior column generation based approaches do not consider the availability, contain numerous modelling issues and are incapable of satisfying the diverse SLA constraints expected in 5G use cases.

The decision for solving the problem of Corr-STN SC, was motivated by the lack of any literature within APS containing solutions that are robust to correlations in the uncertain parameters. The problem of PSTN SC has been tackled in the past by a variety of different authors. However all prior solutions assume that the uncertain durations are stochastically independent. Many sensitive applications contain durations which are correlated, for example industrial production scheduling [25] and scheduling of wind turbines in energy networks [26]. Assuming independence offers no guarantee of finding the most robust schedule if correlations are encountered when the schedule is executed in the real world. However, considering correlations across action durations results in a formulation containing multivariate distributions which are non-linear. Non-linear functions pose additional challenges which can not be handled by conventional LP techniques. By drawing analogies to the probabilistic programming problems studied by Prèkopa and colleagues [27], it was found that the problem was convex, enabling application of convex optimization techniques. Column generation has been applied to such problems in the past [28, 29] and has been shown to work well. One of the key motivators for the use of the column generation method, as opposed to other probabilistic programming techniques is that the constraining factor in terms of efficiency is the dimensionality of the multi-variate distributions considered. Often correlations only exist across subsets of actions. By using column generation we are able to solve a number of much smaller sub problems (one for each set of actions containing a correlation) as opposed to one much larger sub problem. For a more in depth review of techniques for probabilistic programming we refer the reader to Section 2.4.2.

We would like to point out that the two problems tackled in this thesis are very different: both in class (one is convex while the other is combinatorial) and application (one is a stochastic scheduling problem while the other is a telecommunications planning problem). We hope that this highlights the versatility of the technique. Initially it was envisioned that the VNF-PRP would be a practical use case for the Corr-STN work. An intelligent network slice management function requires both planning (as per Chapter 4), and scheduling of actions to adjust the VNF placement and routing in response to predicted SLA violations. These SLA violation events are temporally uncertain and correlated due to factors such as traffic density. The scheduling problem could be solved by using a Corr-STN (as per Chapter 5). This problem is not addressed in this thesis but is outlined in the future work, Section 6.2 of the conclusion.

1.4 Contribution

In Chapter 4, we introduce the first VNF placement model capable of computing a bounded optimal solution containing all of the following features: 1) a valid placement of VNFs to compute nodes, 2) a full routing solution containing a set of valid routing paths and the fraction of traffic to route down each path and 3) the number of VNF replicas required. Similarly, our model is the first that we are aware of, that is capable of satisfying the throughput, latency and availability constraints required of 5G network slices. We show that all of these features can be modelled via a column generation procedure, in which the CGP is a constrained shortest path problem on an augmented network. This allows efficient solutions to be computed to practical sized problems. We experimentally validate this claim on a realistic Mobile Edge Cloud (MEC) test case and show that our model can typically find near optimal solutions within a reasonable time.

In Chapter 5, we introduce and formally define for the first time the Corr-STN, which expands the state of the art on PSTNs by permitting modelling of correlations involving multiple uncertain action durations. We show that the problem of Corr-STN SC can be expressed as a convex one. This enables the application of efficient, optimal solution techniques. We describe the first algorithm in the literature for solving the problem of Corr-STN SC based on column generation. This approach is capable of finding the optimal schedule, in addition column generation is an any-time algorithm allowing us to trade-off numerical time spent with an acceptable optimality guarantee. We present an experimental validation using a drone scheduling test domain with the following findings: 1) prior PSTN SC algorithms assuming independence are not guaranteed to give a conservative estimate of robustness if correlations are encountered, 2) prior PSTN SC algorithms using Boole's inequality are conservative but can be grossly inaccurate and 3) by considering correlations our model is capable of computing more robust schedules versus prior PSTN SC approaches.

Chapter 2

A Survey of Robust Planning and Scheduling

"Uncertainty is the only certainty there is, and knowing how to live with insecurity is the only security."

– John Allen Paulos

2.1 Introduction

In this chapter, we review literature related to robustness in planning, scheduling and optimization. We start by introducing the basic problem definitions. We then look at robustness in APS and present a survey and classification of past approaches. Following this, we review stochastic optimization and discuss some techniques for solving stochastic optimization problems.

Throughout this chapter, we discuss our perspective on these approaches, in particular we motivate the need for APS models with robustness guarantees: some quantifiable measure of the probability that the solution will function as expected. This leads us towards stochastic optimization based approaches and subsequently motivates the application of the column generation method.

We note that the literature in this section is intended to provide a broad perspective on a number of related fields. As a result, we have included a variety of references which allow the reader to gain a more detailed understanding of specific areas. For planning, we refer the reader to the text book by Ghallab et al. [30] and for scheduling, the textbook by Pinedo is an excellent resource [31]. For a general overview of different optimization problems and techniques for solving them we refer the reader to the book by Sinha [32]. Finally, for a more comprehensive review specific to stochastic optimization we refer the reader to Prèkopa [33].

It's also worth pointing out that past surveys of robustness in planning and scheduling are available but they tend to focus on a narrower scope. Vermaelen et al. [34] focus on probabilistic planning and scheduling, Bensalem et al. [35] survey verification and validation techniques, Verderame et al. [36] look at planning and scheduling under uncertainty from an industrial engineering perspective and a number of authors survey scheduling under uncertainty [37, 38]. We would like to stress, that in our survey of robustness, we are referring to robustness as a design requirement (maintaining functionality in the presence of uncertainty), as opposed to a purely probabilistic setting where uncertainty is modelled as an input. Hence, our review covers both deterministic and probabilistic approaches. As far as we are aware, we are the first to present a survey and classification related to robustness in planning, scheduling and optimization, considering both deterministic and probabilistic approaches.

An in depth review of the literature related to the specific problems tackled in this thesis are provided in the relevant chapters. For the VNF-PRP we refer to Section 4.3 and for Corr-STN SC we refer to Section 5.3.

2.2 Background

2.2.1 Mathematical Optimization

Optimization problems involve selecting the best set of resources in order to maximize reward, sometimes subject to constraints. *Mathematical optimization* is the scientific field that studies the mathematical formulation of optimization problems, whereas *mathematical programming* refers to the set of techniques used to solve them.

From a mathematical perspective, the set of resources we wish to select when solving an optimization problem are the *decision variables*, which can be assigned a particular value. The allowable values of the decision variables are constrained by a set of equality or inequality constraints which define the feasible solution space; while the level of reward obtained by selecting a particular set of decision variables is modelled via a mathematical function known as the *objective function.* By solving a mathematical optimization problem, we seek to find an assignment of a feasible value to each of the decision variables, that maximizes or minimizes the objective function. More formally, a mathematical optimization problem is defined as per (2.1):

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & g_i(\boldsymbol{x}) \leq 0, \quad i = 1, 2, \ldots, m \\ & x_i \quad \geq 0, \quad i = 1, 2, \ldots, n \end{aligned} \tag{2.1}$$

Where $\boldsymbol{x} \in \mathbb{R}^n$ is a vector of decision variables, $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function and $g_i(\boldsymbol{x}) \leq 0$ are the constraints that define the set of feasible values for $\boldsymbol{x}$. A solution to a mathematical optimization problem is an assignment of a feasible value to $\boldsymbol{x}$ which minimizes (or maximizes) f.

Mathematical optimization problems are typically classified according to the type of decision variables and class of mathematical functions present. Each class has its own particular suite of solution methods, some classes being more challenging to solve than others. The most basic class occurs when the decision variables are continuous and the constraints and objective function are linear. Such problems are denoted Linear Programs (LP). When the variables are continuous, but the constraints and/or the objective function is non-linear the problem is classed as a Nonlinear Program (NLP). NLPs are significantly more challenging as there is no guarantee that the solution will be optimal. The exception to this is when the functions are nonlinear but convex, in which case efficient algorithms exist capable of computing the optimal solution. Many practical optimization problems feature variables whose values are discrete. These types of problems are prevalent in planning and scheduling applications, for example when finding a plan we cannot take parts of each action, we must select either the whole action or not take it at all. Such a variable is discrete and optimization problems containing discrete variables are known as Mixed Integer Programs (MIP).

Techniques for solving mathematical optimization problems vary drastically depending on the application. We only discuss in detail techniques for certain classes of problems, for a more detailed survey we refer the reader to a relevant textbook [32]. We do mention however, that optimization problems are typically solved using either exact, heuristic or approximation methods. Exact methods can compute the optimal solution but tend to suffer issues with scalability: whereas heuristic methods can often find high quality solutions within a reasonable time but offer no guarantee of solution quality. Approximation methods strike a balance between the two and are often capable of finding a high quality solution in an efficient manner, while providing numerical guarantees on solution quality. Approximation algorithms provide an improved understanding of the problems and underlying structures and can possibly be incorporated within a larger framework. However, they may not be suitable for a direct implementation for practical problems. The guarantees may not hold for the application at hand or the algorithms themselves may not be scalable. If the problem structure is of a particular form, decomposition methods allow one to exploit it and break large scale problems into more manageable problems which can be solved more efficiently.

2.2.2 Planning

Since the conception of AI, planning has been seen as an essential function of intelligence and a key component necessary to develop intelligent agents which can act rationally [39]. Generally speaking, planning can be thought of as the process of deliberating before acting.

A deterministic task planning domain contains a set of *actions* which can be performed in the environment. Actions can only be performed within a state if a number of *preconditions* hold. After an action is performed, a number of conditions change, these are referred to as the *effects* of the action. More formally, a planning domain can be defined as a state transition system:

Definition 1 (State Transition System)**.** *A state transition system is a tuple, $\Sigma = (S, A, \gamma)$, such that S is the set of states, A is the set of actions that can be performed and γ is the state transition function $\gamma : S \times A \rightarrow S$, which defines which actions can be legally applied in which states. If $\gamma(s, a) \neq \emptyset$, then $a \in A$ is applicable in $s \in S$, and will transition the state from s, to some other state $s' \in \gamma(s, a)$.*

In a planning problem, we start with an *initial state* s_0, and want to find a *plan*: $\pi = \langle a_1, a_2, \ldots a_n \rangle$, a sequence of actions which, when applied to our initial state, satisfies a number of goal conditions g. If we denote $S_g \subseteq S$ as the set of goal states that satisfy g, we can formally define a planning problem:

Definition 2 (Planning Problem)**.** *A planning problem is a tuple, $P = \langle \Sigma, s_0, S_g \rangle$, such that Σ is the state transition system, s_0 is the initial state and S_g is the set* *of goal states. The solution to P is any plan π, such that $\gamma(s_0, \pi) \in S_g$.*

Planning problems are typically modelled using the Planning Domain Definition Language (PDDL). PDDL was first introduced by Aeronautiques et al. [40] and was extended in version 2.1 by Fox and Long to include temporal planning containing numerous additional features such as durative actions (actions which have durations) and metrics [41]. We formally define a PDDL temporal planning instance as per Fox and Long:

Definition 3 (PDDL Temporal Planning Instance)**.** *A PDDL temporal planning instance is a tuple, $P = \langle Dom, Prob \rangle$. The domain $Dom = \langle Ps, Fs, A, arity \rangle$, consists of a set of predicate symbols Ps, function symbols Fs, actions (durative or non-durative) A and arity, a function mapping each symbol to their respective arities. In temporal planning, durative actions contain a duration and the conditions and effects can occur at either: the start of the action, the end of the action or for the overall action duration. The problem $Prob = \langle O, s_0, S_g \rangle$, is composed of a set of objects O, initial and goal states s_0 and S_g as per Definition 2.*

Each state in a PDDL temporal planning instance is comprised of three parts: the time of the state s_t, the logical part s_l, containing the set of facts that hold True in the state, and the numeric part s_n, defining the function values in the state at the given time.

Definition 4 (Plan State)**.** *The set of propositions $PROP$, is the set composed of applying the predicate symbols from Ps, to an ordered set of objects $o \subseteq O$, while respecting arities. Likewise, the set of primitive numeric expressions PNE, is composed by applying the function symbols from Fs, to the objects O. The state $s = \langle s^t, s^l, \mathbf{s}^n \rangle$ is a tuple, such that $s^t \in \mathbb{R}$ is the time of the state, $s^l \subseteq PROP$ is the set of propositions in the state and $\mathbf{s}^n \in \mathbb{R}^{|PNE|}$ is a vector defining the values of the PNEs in the state.*

We will illustrate these concepts with reference to a toy domain: the vehicle delivery domain. In this example, the set of objects O is composed of a vehicle v, a package p and two locations $l1$ and $l2$. We refer to the package and vehicle as *locatables* since they can be at a given location. The set of predicates are that the locatables can be *at* a given location: (*at ?loc - locatable ?l - location*), two locations can be connected by a *road*: (*road ?l1 - location ?l2 - location*) and the package can be *in* a vehicle: (*in ?p - package ?v - vehicle*). For simplicity, we do not include function symbols. Finally, in the domain we have three durative actions: the vehicle can *drive* between two locations, it can *pick-up* the package from a location and it can *drop-off* the package at a location.

For the pick-up action to be performed, the vehicle must be at the location for the overall action duration, while the package should be at the same location at the start of the action. Once the action has been performed, the effect is that the package is no longer at the location at the start of the action and the package is in the vehicle at the end of the action. It is assumed that it takes a constant duration of 1 for the vehicle to pick up the package. This action is described using PDDL in Listing 2.1.

For the vehicle to be able to drive between two locations $l1$ and $l2$, the vehicle must be located at the first location $l1$ at the start of the action and there must be a road connecting the two locations for the overall action duration. Once the action has been performed, the effect is that the vehicle is no longer at location $l1$ at the start of the action, and it is at location $l2$ at the end of the action. We assume that the action of driving between two locations takes a constant duration of 5. This action is described using PDDL in Listing 2.2.

For the vehicle to be be able to drop off the package at a location, the vehicle must be at the location for the overall action duration and the package must be in the vehicle at the start of the action. Once the action has been performed, the effect is that the package is no longer in the vehicle at the start of the action and the package is at the location at the end of the action. We assume that this action also takes a constant duration of 1. This action is described using PDDL in Listing 2.3.

Listing 2.1: PDDL pick-up action.

```
(:durative-action pick-up
    :parameters (?v - vehicle ?l - location ?p - package)
    :duration (= ?duration 1)
    :condition (and (over all (at ?v ?l)) (at start (at ?p ?l)))
    :effect (and (at start (not (at ?p ?l))) (at end (in ?p ?v)))
)
```

Listing 2.2: PDDL drive action.

```
(:durative-action drive
    :parameters (?v - vehicle ?l1 ?l2 - location)
    :duration (= ?duration 5)
    :condition (and (at start (at ?v ?l1)) (over all (road (?l1 ?l2))))
    :effect(and (at start (not (at ?v ?l1))) (at end (at ?v ?l2)))
)
```

Listing 2.3: PDDL drop-off action.

```
(:action drop-off
    :parameters (?v - vehicle ?l - location ?p - package)
    :duration (= ?duration 1)
    :condition (and (over all (at ?v ?l)) (at start (in ?p ?v)))
    :effect(and (at start (not (in ?p ?v))) (at end (at ?p ?l)))
)
```

An example proposition is formed by applying the *at* predicate to the vehicle v and location $l1$ with arity 2: (*at v l1*). Thus we can define the logical part of the initial state as the set of propositions: $s_0^l = \{$(*at v l1*), (*at p l1*), (*road l1 l2*)$\}$. The initial state begins at time 0, since there are no function symbols, the initial state is defined as $s_0 = \langle 0, s_0^l, \emptyset \rangle$. The goal is that the package p should be located *at* location $l2$. This can be defined as the single proposition (*at p l2*).

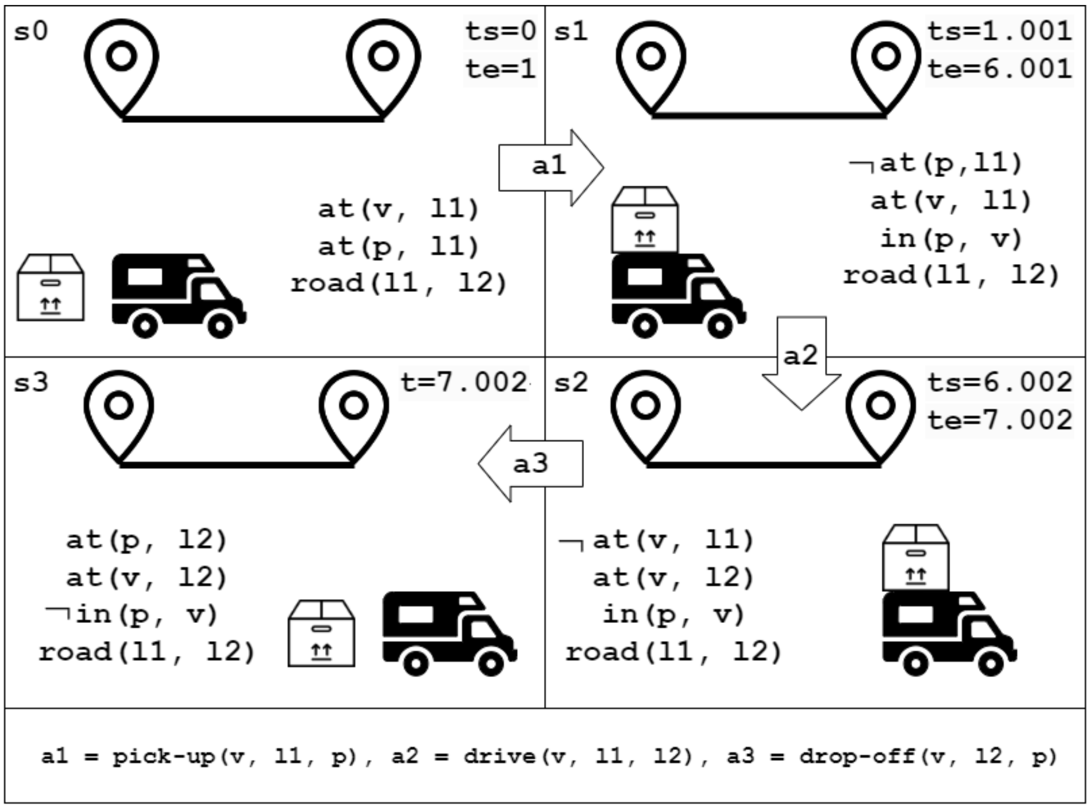


Figure 2.1: Example plan for vehicle delivery problem.

An example plan consists of applying three actions sequentially:

$$\pi = \langle \text{pick-up}(v,\ l1,\ p),\ \text{drive}(v,\ l1,\ l2),\ \text{drop-off}(v,\ l2,\ p)\rangle$$

The resulting state transition is explained visually in Figure 2.1. We show the conditions at the start of each action. The start and end times of the action are described as ts and te respectively. Note that to distinguish between the state at the start and end of the action, we add a small separation of 0.001 seconds, hence $a1$ finishes at $te = 1$, whereas $a2$ begins at $ts = 1.001$. Here, applying the plan transitions the state from the initial state s_0, to a goal state s_3.

2.2.3 Scheduling

Pinedo [31] defines scheduling as the problem of assigning resources (for example time, computer processors or industrial machinery) in order to complete tasks over a period of time. In this thesis, we look at one particular application of scheduling which involves assigning times to actions from a plan. Simple Temporal Networks (STN), first introduced by Dechter et al. [42], are graphs used to model such problems. An STN is a graph in which the nodes correspond to *time-points* and the edges (*links*) correspond to durations between the time-points.

Definition 5 (STN). *An STN is a tuple, $S = \langle T, C \rangle$ where $t \in T$ is the set of time-point vertices and C is the set of temporal requirement constraints or edges between two time-points; normally written in the form $c(t_j, t_i) = t_j - t_i \in [l_{c,ij}, u_{c,ij}]$, where $l_{c,ij}$ and $u_{c,ij}$ are the lower and upper bound on the allowable duration between the time-points t_i and t_j. Let $s(t) \in \mathbb{R}^+$ be the assignment of a real value to the time point. A schedule s is the assignment $s(t)$ for all $t \in T$, while a valid schedule, is one in which $c(t_j, t_i) \in [l_{c,ij}, u_{c,ij}]$ for all $c \in \mathcal{C}$. An STN with at least one valid schedule is a consistent STN.*

As an example, imagine a student tasked with completing a project by a professor. The student has been given a deadline of 10 days from the current date to submit their project. They know that the project will take 6 days to complete and they must decide when to work on the project such that they meet the deadline. Such a problem can be modelled by the STN provided in Figure 2.2. The time-points represent the following events:

t_1 : Time at which the professor gives the project to the student.

t_2 : Time at which the student begins work on the project.

t_3 : Time at which the student finishes work on the project.

t_4 : Time of the deadline (when the student must finish the work by).

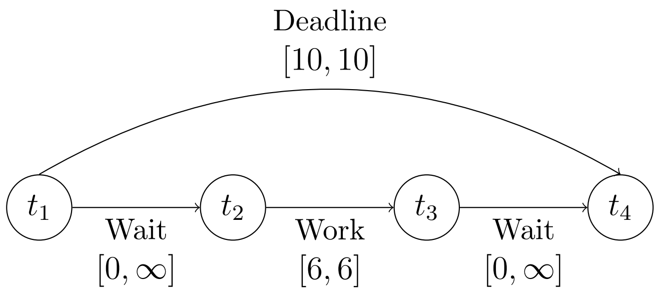


Figure 2.2: Example Simple Temporal Network.

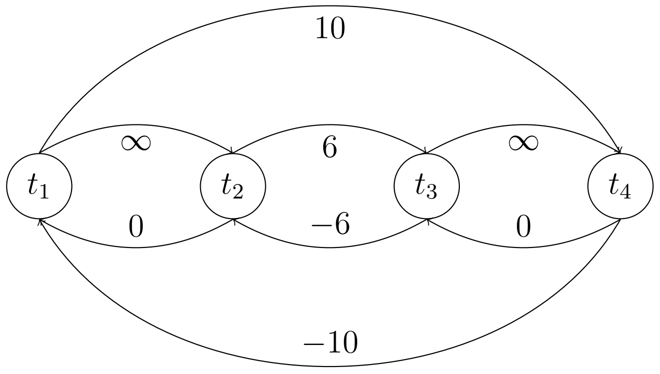


Figure 2.3: Digraph representation of STN from Figure 2.2

While the constraints $c(t_3, t_2) = t_3 - t_2 \in [6, 6]$, represents the action of the student working on the project; $c(t_4, t_1) = t_4 - t_1 \in [10, 10]$ represents the deadline; and the two other edges: $c(t_2, t_1) = t_2 - t_1 \in [0, \infty]$ and $c(t_4, t_3) = t_4 - t_3 \in [0, \infty]$ represent times at which the student is idle. Such is typically the case, the student may choose to wait as long as possible before beginning work on the project. In which case $s = \{t_1 := 0, t_2 := 4, t_3 := 10, t_4 := 10\}$, represents a valid schedule.

It should be noted that we can represent the constraints $t_j - t_i \in [l_{c,ij}, u_{c,ij}]$ in the form of two less than inequalities: $t_j - t_i \leq u_{c,ij}$ and $t_i - t_j \leq -l_{c,ij}$. If we represent each of these constraints as an edge in a graph, we can present the STN as a directed graph with edge weights as shown in Figure 2.3.

Dechter et al. [42] show that checking for negative cycles in the digraph is equivalent to checking consistency: if there are no negative cycles, then there is at least one valid schedule. This enabled the application of efficient shortest path algorithms such as Floyd-Warshall to check consistency and find a schedule to the STN [43].

2.3 Robust Planning and Scheduling

In this section we review literature related to *how to achieve robustness* in APS. We have identified three key groups of approaches: *proactive*, *reactive* or *probabilistic*. Proactive measures attempt to consider in advance the possible failure sources and develop the solution accordingly, whereas reactive measures react to the failures as they arise. On the other hand, probabilistic approaches explicitly reason over the probability of potential failures.

We further group proactive measures according to verification and validation, contingency based and redundancy based. Vieira et al. present a comprehensive survey on reactive strategies for scheduling and classify approaches as either: dynamic or predictive/reactive [44]. We use the same classification in this survey. Finally, we separate probabilistic approaches into static and dynamic, whereas static approaches find a single solution, dynamic approaches compute a policy.

2.3.1 Proactive Approaches

Verification and Validation Verification and validation are two techniques which can be used to check that the model behaves *as expected*.

In planning, verification and validation has been applied to verify domain models [35] as well as plans [45] and planners [46, 47]. Verification can be completed by either model checking [48] or model testing [49, 50]. The prior aims to ensure that the model works for all possible states, while the latter generates a set of test cases to ensure that no errors are present. Raimondi et al. note that model testing is often preferable to model checking, since the size of the state space may be prohibitive (it is not possible to test exhaustively all states for failure) and the domain model may contain features which are not possible to encode in the language of the model checker [49]. On the other hand, model testing is reliant on the test cases covering a sufficient portion of the state space; model checking can prove the absence of faults, whereas model testing can only show whether faults are present in the test cases [35].

Contingency Based Approaches As opposed to a single plan or schedule, contingency based techniques produce a solution with multiple branches, usually in the form of decision tree, where each branch is a valid solution for one possible uncertain outcome.

Contingency based planning was introduced by Goldman et al. [51] and extended to include sensing actions which could be used to gather information about the state that the agent is in [52,53]. Even with information from sensors, contingent planners need to deal with incomplete information and partial observability: the state that the agent thinks it is in, may not be the state that the agent is actually in. Bonet and Geffner modelled contingent planning as a search in the space of *belief states* [54]. In this approach, the belief space is the space of all possible states, which is exponential in the number of states in the domain [55]. A number of planners have been proposed which pose the problem of contingent planning as a search on a binary decision diagram in the belief space [55–58]. More recent advancements involve compiling the contingent problem into a non-deterministic search in the state space as opposed to the belief space [59,60], thus enabling the application of classical planning techniques. This result has been used to compute offline solutions to contingent planning problems [61, 62] and extended to include temporal planning problems [63]. The problem of contingent planning is similar in a sense to the problem of *just in case* scheduling [64], in which contingent schedules are computed to cover the failures most likely to be encountered.

Redundancy based Approaches Rather than explicitly modelling all possible contingencies, redundancy based techniques aim to absorb some of the uncertainty by introducing conservatism in the form of redundancies, thus mitigating against potential failures. As such these approaches could also be referred to as *mitigation* based approaches.

In a scheduling context, redundancy has been incoporated by adding time to operations, or inserting dummy operations to cover delays in the event of faults. Mehta [65] and O'Donovan et al. [66] add idle time to account for failures in the problem of single machine scheduling. Davenport et al. [67] add two new measures of temporal slack: the first ensures that each activity will have a set amount of slack, while the latter varies the amount of slack provided to the activity depending of the whereabouts of the activity in the schedule. Intuitively, later activities require more slack since there are more opportunities for a failure upstream of the activity. By introducing the slack in the scheduling problem definition, as opposed to simply extending the activity durations, their approach allowed reasoning over *where* to put the slack. Scheduling with redundancy can be compared to the problem of Strong Controllability (SC) in Simple Temporal Networks with Uncertainty (STNU), whereby a schedule is found for all possible outcomes of the uncertain action durations. This problem is covered in Section 5.2. Cimatti et al. [68] extend SC to *strong* temporal plans as opposed to schedules. Strong temporal planning seeks to find a plan that will succeed regardless of the outcome of uncertain action durations.

2.3.2 Reactive Approaches

Predictive/Reactive Approaches In predictive/reactive approaches, a proactive plan or schedule is computed in advance which is then adjusted in the event of a failure.

In scheduling, reactive approaches typically involve *rescheduling* in the event of disruption. This disruption can be caused by a number of factors, for example, machine failure, job cancellation, new jobs or lack of resources [69, 70]. In order to repair the schedule, a number of techniques have been proposed: shifting the start time of all tasks in the schedule by some fixed value (right shift rescheduling) [71], repairing only the tasks which are affected by the disruption (partial rescheduling) [72] and complete regeneration of the schedule for all tasks not yet completed (schedule regeneration) [73].

A similar line of reasoning has been undertaken within the planning community. Reactive measures for dealing with plan failure were discussed by Fox et al. [74]. In particular, they identify two different approaches: replanning and plan repair. The two are contrasted through the notion of plan *stability*: a measure of how different a new plan is from the existing one. In replanning, a completely new plan is generated with no consideration paid to how different it is from the existing one; whereas in plan repair the aim is to modify the existing plan to work in the new situation, while minimizing the changes. Many techniques have been proposed in literature for replanning and plan repair [75, 76].

Dynamic Approaches In dynamic approaches, the solution is not a schedule or plan, but a strategy, which is used to select resources or actions online.

In dynamic scheduling, jobs are assigned as the resources become available according to some preference criteria called a *priority rule* or dispatch rule (e.g. shortest processing time), a survey of which is provided by Panwalkar and Iskander [77]. This shares similarities with the problem of dynamic controllability of

STNUs. An STNU is dynamically controllable if it is possible to compute a dispatch strategy which can be adapted in real time based on the the observation of past events. A number of algorithms have been proposed for proving dynamic controllability of STNUs and generating dispatch strategies [78–80].

Rather than committing to a plan that is then adjusted upon failure, Do et al. produce a flexible, partially ordered plan enabling actions to be reordered during execution [81]. Lima et al. relax the causal structure in temporal plans, also resulting in a partially ordered plan [82]. This partially ordered plan is then used to enumerate a set of complete plans, where each such plan has an associated probability of success. The probabity of success of each plan is reasoned over to select the best action at execution time. Orlandini et al. present a similar approach which automatically synthesizes a plan execution controller based on the generation of a winning strategy in timed game automata [83]. Fox et al. [84] learn a policy through solving deterministic plan instances, and then passing the plans as input to a decision tree classifier.

2.3.3 Probabilistic Approaches

In the previous sections, most of the planning and scheduling models were deterministic. Robustness in these approaches is achieved through either anticipating uncertainty and taking proactive measures, or observing the uncertainty and taking reactive measures. Uncertainty was not modelled quantitatively. Such approaches are outlined in this section.

Probabilistic Dynamic In probabilistic planning, the probability of reaching a state, given an action is explicitly encoded in the state transition system [85]. This is achieved through modelling the planning problem as a Markov Decision Process (MDP) [86] which can be solved through policy or value iteration [87]. MDPs differ from traditional planning techniques in that their solution is not necessarily a plan but a *policy* defining which action to take in which situation. The goal of MDPs is typically to find a policy which maximizes the reward function, where high rewards are generally applied to states that achieve some goal conditions, thus mapping the goal conditions onto the MDP [88]. MDPs have been extended to handle a variety of different features: for example partial observability [89], uncertainty in transition probabilities [90], cost and resource constraints [91], chance constraints [92] and path constraints [93]. A number of extensions to PDDL have also been proposed to model planning problems as MDPs [94, 95] for which support is provided in a number of planners [96, 97].

Probabilistic Static MDPs require discretization of time and resources to model uncertainty which can result in a combinatorial explosion of the state space. In contrast, Beaudry et al. [98] use random variables to model uncertainty. Their approach employs forward chaining search, coupled with a Bayesian network which maintains dependency relations between the random variables. The Bayesian network is generated dynamically as the actions are applied in the search and is used as input to a Bayesian network inference algorithm based on sampling, which computes the probability of success of the returned plan. Since the resulting plan is often conservative, Coles [99] extends the work of Beaudry et al. to consider soft goals (goals which can be violated at the expense of a penalty paid) and opportunistic branches (branches added to the plan which can be taken advantage of if random variable values are better than expected).

Probabilistic planning deals with finding a policy, while considering the probability that a state will be reached *after* taking an action. However these techniques do not reason over the uncertainty in action duration. This reasoning is typically handled in a scheduling context. Probabilistic Simple Temporal Networks (PSTN) are an extension of STNs in which the duration of actions and events are modelled using probability distributions [100]. A number of papers have been written on the topic of finding a schedule to a PSTN while maximizing the probability that the schedule will succeed [100–102] or minimizing cost, subject to a tolerance on the probability of success [13, 103]. PSTNs form the backbone of the work completed in Chapter 5 of this thesis and so a comprehensive review of such approaches is provided in Section 5.3.

2.4 Optimization under Uncertainty

In this section, we discuss approaches to dealing with uncertainty from an optimization perspective. We focus our attention on stochastic optimization, in which the uncertainty is described with the use of random variables. We introduce the problem definition and review and discuss some techniques used to solve these problems.

2.4.1 Stochastic Optimization

Stochastic optimization covers any optimization problem in which random variables are present. In this thesis, we look at two particular cases. In the first case, *probability maximization*, we want to maximize the probability that the constraints are satisfied. Probability maximization problems can be written in the general form:

$$\begin{array}{lll} \max\limits_{\boldsymbol{x}} & P(f_1(\boldsymbol{x},\boldsymbol{\xi}) \geq 0, \dots, f_k(\boldsymbol{x},\boldsymbol{\xi}) \geq 0) & \\ \text{subject to} & g_i(\boldsymbol{x}) & \leq 0, \quad i = 1, 2, \dots, m \\ & x_i & \geq 0, \quad i = 1, 2, \dots, n \end{array} \tag{2.2}$$

In this case, the decision variables x and constraint functions g are as per Section 2.2.1. Here we want to maximize the probability that the constraints $f_1(\boldsymbol{x},\boldsymbol{\xi}) \geq 0, \dots, f_k(\boldsymbol{x},\boldsymbol{\xi}) \geq 0$ are satisfied, where $\boldsymbol{\xi}$ is a multivariate random vector with some probability distribution.

In the second case, *chance-constrained*, we want to satisfy the constraints with some tolerance on risk. A chance constrained optimization problem can be written in the form:

$$\begin{array}{lll} \min\limits_{\boldsymbol{x}} & f(\boldsymbol{x}) & \\ \text{subject to} & P(g_1(\boldsymbol{x},\boldsymbol{\xi}) \geq 0, \dots, g_m(\boldsymbol{x},\boldsymbol{\xi}) \geq 0) \geq p & \\ & h_i(\boldsymbol{x}) & \geq 0,\ i = 1, 2, \dots o, \\ & x_i & \geq 0,\ i = 1, 2, \dots, n \end{array} \tag{2.3}$$

where we want to minimize a cost function f subject to the fact that the constraints $g_1(\boldsymbol{x},\boldsymbol{\xi}) \geq 0, \dots, g_m(\boldsymbol{x},\boldsymbol{\xi}) \geq 0$ must be satisfied with some probability $p \in \{0, 1\}$.

Charnes et al. [104] made one of the first attempts at formalising stochastic optimization problems through addressing the problem of scheduling oil supplies subject to uncertainty from weather and demand. Later work by Charnes and Cooper formally defined the chance constrained optimization problem [105]. It should be noted, that the work of Charnes and Cooper focused on the case that the chance constraints were treated separately: $P(g_i(\boldsymbol{x},\boldsymbol{\xi}) \geq 0) \geq p_i$, where p_i is defined for each constraint g_i, as the required probability that it is satisfied.

Prèkopa made the generalization to the case of multi-variate distributions [106] and joint chance constraints as per (2.3). The author extended this work [107–109], where it was observed that chance constrained problems dealing with log-concave probability measures are convex programs. Extensive theory and efficient solution techniques were developed following this seminal work [110–113].

2.4.2 Solution Methods

Stochastic optimization problems of the form provided in (2.2) and (2.3) have been solved using a variety of techniques. Prèkopa [106] provided convergence proofs for the method of feasible directions [114], and used it to solve the problem of energy production planning under uncertainty [115]. In the method of feasible directions, two sub problems are solved iteratively: the first finds a direction which is feasible for a sufficiently small step size, while the second computes the maximum step size reducing the objective, for which the direction remains within the feasible region.

In the sample average approximation technique, the probabilistic constraint or objective is replaced by a sample average approximation of the random variables. The resulting deterministic optimization problem can then be solved. Luedtke and Ahmed [116] solve a problem equivalent to (2.3) and derive sample size conditions required to have high confidence that the solution to the approximation will provide a valid lower bound and feasible solution.

Cutting plane methods [117] are techniques which can be used to form an outer approximation of the convex set formed by the probabilistic constraints, using a set of linear constraints called cutting planes. In this approach, the intersection of the cutting planes forms a polytope approximating the feasible region, which can be solved as an LP. However the solution to the LP is not guaranteed to be feasible to the original problem. The outer approximation is refined iteratively by adding new constraints to the LP until the optimal solution is feasible. Prèkopa et al. [118] use the cutting plane method of Kelley [117], in which the generated cuts are not guaranteed to support the feasible region [119]. In order to improve the efficiency of the approach, Veinott [120] proposed using a line search between a slater point (a known point within the feasible set) and the optimal solution to the master problem to generate a tighter cut. The resulting supporting hyperplane-method has been used by Arnold et al. [121] and Szántai [122] in the context of stochastic optimization.

Prèkopa [123] showed that if the probabilistic constraint is separable, then the random variable can can be replaced with a set of points satisfying the probabilistic constraint, referred to as p-efficient points. Prèkopa [118] later presented a method for solving stochastic optimization problems with discrete random variables which relied on explicit enumeration of all such points. Rather than explicitly enumerating all p-efficient points, Dentcheva et al. [124] use column generation to generate the points iteratively in their so-called cone-generation method.

Fabian et al. [29] tackle a problem of the form present in (2.2). Their approach forms an inner approximation of the convex set formed by the probabilistic constraint using a number of approximation points, and then uses column generation to iteratively refine the inner approximation. Their approach is similar to the cone generation method, however the probabilistic constraint is approximated in the master problem via the inner approximation. The result is that each new column is not required to satisfy the probabilistic constraint and so the sub problem is much simpler. They show that the sub problem amounts to non-linear unconstrained minimization which can be solved via gradient descent.

2.5 Discussion and Conclusions

In reviewing techniques related to robust APS, a number of approaches were identified.

Verification and validation can help to rule out modelling errors and ensure that the planning and scheduling model is behaving *as expected*. Even if the solution is valid with respect to the model, it may be the case that the plan or schedule fails during execution due to discrepancies in the model versus the execution environment. In this case, robustness can normally be achieved using proactive or reactive measures.

By considering and planning for possible faults in advance, proactive measures can prevent delays at execution time. Redundancy based approaches are simple and easy to implement, however they can be overly conservative. Using contingency based approaches can be beneficial as they offer a solution that will work for every outcome considered, without needing to recompute a new solution on line. Since each uncertain outcome in contingent planning and scheduling results in a new branch in the decision tree, these approaches are mainly suited to applications in which the space of potential outcomes is small. In reality, there are infinitely many potential outcomes that can arise in the real world. Therefore, using contingency based techniques requires careful consideration of the most likely sources of failure. In some cases, it may not be computationally feasible to plan for all possible contingencies.

A different approach may be to develop a solution which can be refined and adapted online at execution time in the event of failure. This allows the model to react to all possible situations that may be experienced in the real world. Repairing or recomputing a plan or schedule at execution time can be disruptive, particularly for time sensitive tasks. Additionally, using a dynamic execution strategy offers no guarantee that the solution being developed is the best one with respect to any desired metric.

All these approaches decouple the planning and scheduling with the computation of robustness and have no way of reasoning over the likelihood of failures. Consequently, such approaches offer no guarantee of finding the most robust solution. As new regulations call for increased transparency and risk awareness in AI systems [8, 125], having numerical guarantees on robustness is becoming increasingly important. For this reason it was decided to address robust APS from a probabilistic perspective. In this thesis, we focus primarily on probabilistic static solutions as opposed to dynamic ones. Such problems can be solved effectively using techniques from the stochastic optimization literature.

In the sample average approximation approach to stochastic optimization, the number of samples required can be prohibitive when the dimension of the data is large [126]. Rather than using random sampling to approximate the distribution, decomposition based approaches such as the cutting plane method and column generation explicitly solve a smaller optimization problem to find the best points (or cuts) to refine the approximation.

Note, that every LP has an associated dual linear program, such that the variables in the *dual* problem are associated with the constraints of the original *primal* problem and vice versa [127]. Column generation and cutting plane methods can be seen as doing the same thing but viewed from a different perspective: generating variables using column generation is analogous to generating cutting planes on the dual problem. The choice of which approach to use is normally driven by modelling convenience, for example which approach permits the simplest sub problems. Cutting plane methods are generally well suited to problems involving a complicated set of variables. Without these variables, or when the variables are fixed the problem becomes separable and simple. Likewise, column generation is normally used when there is a complicating constraint set [128].

When dealing with probabilistic constraints, cutting plane methods form an outer approximation, whereas column generation forms an inner approximation of the feasible set. In column generation, the inner approximation gradually grows within the feasible region, whereas in cutting plane methods, the outer approximation gradually shrinks around the feasible region. As a result, on any iteration, the solution to the approximate problem in column generation is guaranteed to be a feasible solution to the original problem. This is not the case when an outer approximation is used.

For these reasons, it was decided to focus on the column generation method in this thesis. In the next chapter we formally introduce the reader to this method.

Chapter 3

Technical Preliminaries on the Column Generation Method

"Nothing is particularly hard if you divide it into small jobs."

– Henry Ford

3.1 Introduction

In this chapter, we introduce the reader to the column generation method. In particular, we will discuss the basic theory behind it and some of its characteristics. Since this thesis is application orientated and it is not assumed that the reader will be well-versed or even aware of the technique, we will begin with the fundamentals. Most of this chapter will be focused on getting an *intuition* for how it works with the more in depth theory left to other references [129]. In Section 3.2 we discuss the fundamentals of linear programming, the Simplex method and the concept of reduced cost. In Section 3.3 we introduce the theory behind the column generation method and in Section 3.4 we illustrate the practical application of these concepts with reference to the cutting stock problem. For readers already familiar with these concepts, feel free to progress to Chapter 4.

3.2 Linear Programming

To understand the basic principles of column generation, it is important to first discuss the fundamentals of linear programming. LPs are special cases of optimization problems in which all functions and constraints are linear, as per (3.1):

$$\begin{aligned} \max_{\boldsymbol{x}} \quad & \boldsymbol{c}^T \boldsymbol{x} \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \\ & x_i \quad \geq 0, \quad i = 1, 2, \ldots, n \end{aligned} \tag{3.1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the decision variable vector, $\boldsymbol{c}$ is an n-dimensional vector of cost coefficients, $\boldsymbol{A}$ is an $m \times n$ dimensional constraint parameter matrix and $\boldsymbol{b}$ is the m-dimensional vector of bounds.

The feasible region of the LP is bounded by a polytope (see Figure 3.1) formed by the intersection of the constraints. Any solution which lies within the feasible region of the LP is denoted a basic solution, while those which lie on one of the corner vertices of the polytope are *Basic Feasible Solutions* (BFS). Each BFS corresponds to a number of *basic* variables $\boldsymbol{x}_B \subseteq \boldsymbol{x}$ having non-zero value and *non-basic* variables $\boldsymbol{x}_N = \boldsymbol{x} \setminus \boldsymbol{x}_B$, whose value is zero.

Simplex Method The simplex procedure was first introduced by Dantzig [130] and is one of the most widely utilised algorithms for solving LPs. Note that the optimal solution to the LP (3.1) corresponds to one of the BFS vertices of the polytope shown in Figure 3.1. When solving (3.1) using the simplex procedure, the algorithm iteratively moves along an improving adjacent edge of the current vertex of the simplex polytope, to a new vertex, until it reaches the optimal solution. Since each vertex corresponds to a BFS, moving from one vertex to another corresponds to a non-basic *entering* variable entering the simplex basis (i.e. taking non-zero value). The edge to traverse is selected in the *pricing* step of the algorithm, by finding the edge with the highest improvement in the objective function amongst all the adjacent edges (referred to as the steepest edge). The steepest edge corresponds to the one with the greatest *reduced cost*. The BFS vertex corresponding to the optimal solution will have no improving adjacent edge, meaning that traversing the edge will only reduce the objective function and so the algorithm can terminate. This procedure is highlighted in Figure 3.1.

Reduced Cost Note that we can rewrite the inequality in (3.1) as an equality through the inclusion of slack and surplus variables; and that we are free to change the order of the columns and variables. As such any LP can be written in the form of (3.2):

$$\begin{aligned} \max_{\boldsymbol{x}} \quad & \begin{pmatrix} \boldsymbol{c}_B^T & \boldsymbol{c}_N^T \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_B \\ \boldsymbol{x}_N \end{pmatrix} \\ \text{subject to} \quad & \begin{bmatrix} \boldsymbol{A}_B & \boldsymbol{A}_N \end{bmatrix} \begin{pmatrix} \boldsymbol{x}_B \\ \boldsymbol{x}_N \end{pmatrix} = \boldsymbol{b}, \\ x_i \quad & \geq 0, \quad i = 1, 2, \ldots, n \end{aligned} \tag{3.2}$$

At any iteration, since the non-basic variables are zero, the following is a BFS to the above:

$$\boldsymbol{A}\boldsymbol{x} = \begin{bmatrix} \boldsymbol{A}_B & \boldsymbol{A}_N \end{bmatrix} \begin{pmatrix} \boldsymbol{x}_B \\ \boldsymbol{0} \end{pmatrix} = \boldsymbol{b} \tag{3.3}$$

and hence $\boldsymbol{x}_B = \boldsymbol{A}_B^{-1}\boldsymbol{b}$. The equivalent objective value is:

$$\boldsymbol{c}^T\boldsymbol{x} = \begin{bmatrix} \boldsymbol{c}_B & \boldsymbol{c}_N \end{bmatrix} \begin{pmatrix} \boldsymbol{x}_B \\ \boldsymbol{0} \end{pmatrix} = \boldsymbol{c}_B\boldsymbol{A}_B^{-1}\boldsymbol{b} \tag{3.4}$$

We now consider that we are moving to another BFS, such that some non-basic variable assumes a non-zero value, $\boldsymbol{x} = \begin{pmatrix} \bar{\boldsymbol{x}}_B & \bar{\boldsymbol{x}}_N \end{pmatrix}$. Taking the constraint (3.3),

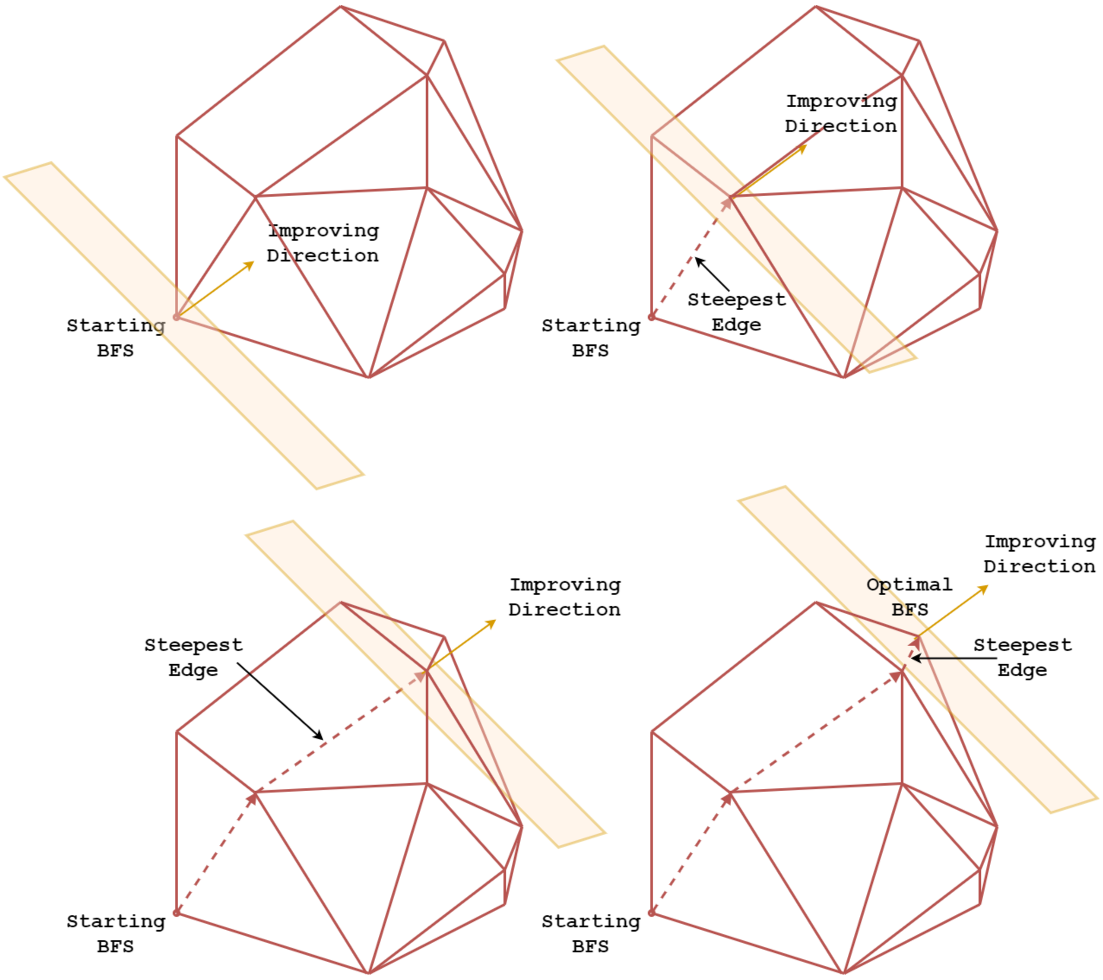


Figure 3.1: Polytope representing feasible region of an LP. The simplex algorithm starts at one of the vertices of the polytope and repeatedly moves along the steepest edge in the improving direction until it reaches the optimal solution. The steepest edge is found by solving (3.8).

we have (3.5):

$$\boldsymbol{A}\boldsymbol{x} = \begin{bmatrix}\boldsymbol{A}_B & \boldsymbol{A}_N\end{bmatrix}\begin{pmatrix}\bar{\boldsymbol{x}}_B \\ \bar{\boldsymbol{x}}_N\end{pmatrix} = \boldsymbol{b} \tag{3.5}$$

Rearranging for the basic variables gives $\bar{\boldsymbol{x}}_B = \boldsymbol{A}_B^{-1}\boldsymbol{b} - \boldsymbol{A}_B^{-1}\boldsymbol{A}_N\bar{\boldsymbol{x}}_N$. The equivalent objective is (3.6):

$$\boldsymbol{c}^T\boldsymbol{x} = \begin{bmatrix}\boldsymbol{c}_B & \boldsymbol{c}_N\end{bmatrix}\begin{pmatrix}\boldsymbol{A}_B^{-1}\boldsymbol{b} - \boldsymbol{A}_B^{-1}\boldsymbol{A}_N\bar{\boldsymbol{x}}_N \\ \bar{\boldsymbol{x}}_N\end{pmatrix} = \boldsymbol{c}_B\boldsymbol{A}_B^{-1}\boldsymbol{b} - \boldsymbol{c}_B\boldsymbol{A}_B^{-1}\boldsymbol{A}_N\bar{\boldsymbol{x}}_N + \boldsymbol{c}_N\bar{\boldsymbol{x}}_N \tag{3.6}$$

If we want to find out how the objective function will change through modifying the basic and non-basic variables, then we can look at the difference in the objective value (3.6) versus (3.4), ρ:

$$\begin{aligned}\rho = \begin{bmatrix}\boldsymbol{c}_B & \boldsymbol{c}_N\end{bmatrix}\begin{pmatrix}\bar{\boldsymbol{x}}_B \\ \bar{\boldsymbol{x}}_N\end{pmatrix} - \begin{bmatrix}\boldsymbol{c}_B & \boldsymbol{c}_N\end{bmatrix}\begin{pmatrix}\boldsymbol{x} \\ \boldsymbol{0}\end{pmatrix} &= \boldsymbol{c}_N\bar{\boldsymbol{x}}_N - \boldsymbol{c}_B\boldsymbol{A}_B^{-1}\boldsymbol{A}_N\bar{\boldsymbol{x}}_N \\ &= \sum_{i\in N}(c_i - \boldsymbol{c}_B\boldsymbol{A}_B^{-1}A^i)x_i\end{aligned} \tag{3.7}$$

Note that $N = \{i \ : \ 1 \leq i \leq n, \ x_i \in \boldsymbol{x}_N\}$, is the index set of non-basic variables and that A^i refers to the column of coefficients from $\boldsymbol{A}_N$ associated with variable x_i.

Intuitively, (3.7) tells us whether the objective function will improve if the BFS was to change. For any non-basic variable x_i whose current value is zero, the term $\rho_i = c_i - \boldsymbol{c}_B\boldsymbol{A}_B^{-1}A^i$ is the amount by which the objective will change for every positive unit increase of x_i, also known as the *reduced cost*. If we are seeking to maximize the objective function (as per (3.1)) and the reduced cost ρ_i is positive, then it means that if variable x_i were to assume a non-zero value, the objective function would increase and so it is an *improving variable*. Note, the term $\boldsymbol{c}_B\boldsymbol{A}_B^{-1}$ is the vector of dual variables $\boldsymbol{y}$ associated with the constraint

$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. In the pricing step of the simplex method, we can find the entering variable by finding the non-basic variable with the greatest reduced cost:

$$\arg\max\{c_i - \boldsymbol{A}^{i^T}\boldsymbol{y} \mid i \in N\} \tag{3.8}$$

If at any iteration, the entering variable is not an improving variable, then the current BFS is optimal and the simplex procedure can terminate. This procedure is discussed in greater detail by Chvatal [131], for more information we refer the reader there.

3.3 Column Generation Method

In many cases, the number of variables can be prohibitively large when solving (3.8) explicitly. The intuition behind the column generation method is that we can instead solve a smaller problem using a subset of the variables (and columns). We then iteratively grow this problem columnwise by adding variables and columns until the problem is big enough that it contains the optimal solution to the original, much larger problem.

Restricted Master Problem From henceforth, we refer to (3.1) as the *Master Problem* (MP) and use $M = \{1, 2, \ldots, n\}$, to refer to the index set of all variables present in the MP. Now assume that we are working with a subset of *generated* variables, such that $G \subseteq M$, is the index set of generated variables. If $\boldsymbol{x}_G$ and $\boldsymbol{c}_G$ are the subvectors of $\boldsymbol{x}$ and $\boldsymbol{c}$ corresponding to the index set G; and $\boldsymbol{A}_G$ is the submatrix of $\boldsymbol{A}$ corresponding to the set of columns in G, we can write a *restricted* version of (3.1), referred to as the *Restricted Master Problem* (RMP), as per (3.9):

$$\begin{aligned} \max_{\boldsymbol{x}} \quad & \boldsymbol{c}_G^T \boldsymbol{x}_G \\ \text{subject to} \quad & \boldsymbol{A}_G \boldsymbol{x}_G \leq \boldsymbol{b}, \\ & x_i \quad \geq 0, \quad i \in G \end{aligned} \tag{3.9}$$

We use $B \subseteq M$, to refer to the index set of optimal basic variables from the MP (3.1). Since all non-basic variables take a value of zero, if $B \subseteq G$, then solving the RMP (3.9), is equivalent to solving the MP (3.1). The non-entered, *unknown* variables corresponding to the index set $M \setminus G$, are treated as non-basic variables taking a value of zero in the current BFS. As such, we can start the RMP with some subset of generated variables and continuously add variables and columns until the set G, contains all the optimal basic variables. The number of generated variables can be significantly less than the number of variables in the MP, making the RMP much easier to solve, while in the worst case the procedure involves solving the same problem (i.e. RMP = MP). An illustrative example of the MP and RMP features is provided in Figure 3.2.

Column Generation Problem This begs the question: how can we know when all of the optimal basic variables have been included in the RMP? We can use the concept of reduced cost introduced previously and solve the following optimization problem called the *Column Generation Problem* (CGP) (often referred to as the oracle or pricing problem):

$$\max\{c_i - \boldsymbol{A}^{i^T} y \mid \boldsymbol{A}^i \in \boldsymbol{A}\} \tag{3.10}$$

where we find a new column $\boldsymbol{A}^i$, that maximizes reduced cost. If the optimal solution to (3.10) is non-positive then we know that the best column is not an improving one and consequently all other columns are also not improving and so the process can terminate.

The structure of the CGP is inherently linked to original MP, often generated

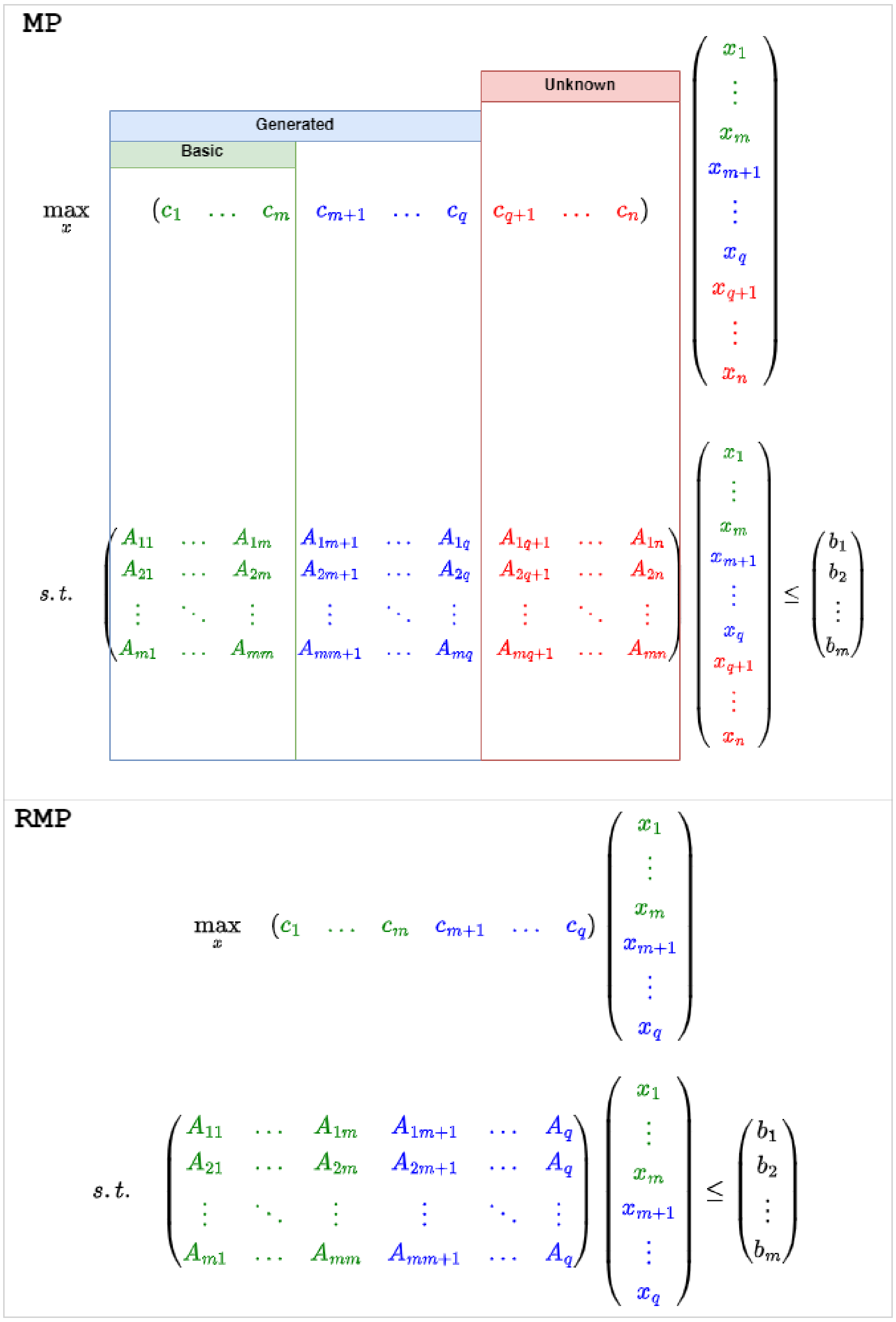

MP
Unknown
Generated
Basic
$\max_x \quad (c_1 \ \dots \ c_m \ c_{m+1} \ \dots \ c_q \ c_{q+1} \ \dots \ c_n)$
$s.t.$
RMP
$\max_x \quad (c_1 \ \dots \ c_m \ c_{m+1} \ \dots \ c_q)$
$s.t.$

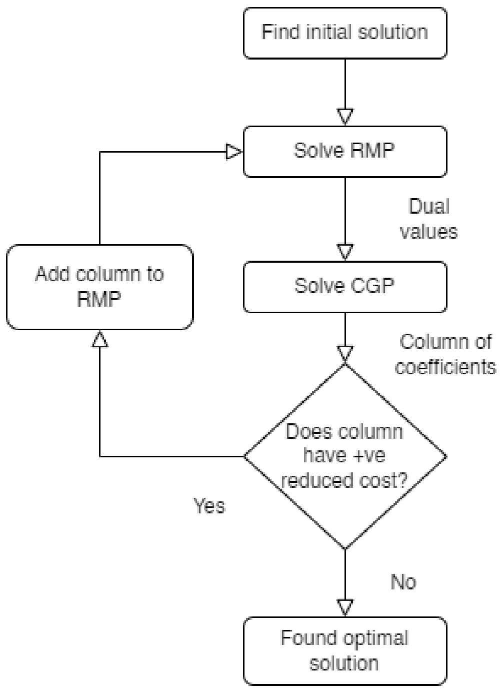
Figure 3.2: Image showing MP for an LP (top), and equivalent RMP (bottom). The RMP is grown columnwise on each iteration by adding columns from the unknown set to the generated set. When the RMP contains all the optimal basic variables, the solution to the RMP is the same as the solution to the MP. Note that $q = |G|$.

columns must abide by a particular structure. As a result, the CGP can take many forms (for example an LP, an NLP or a MIP). Typically, column generation is well suited when the CGP is computationally tractable, since it must be solved in an iterative manner.

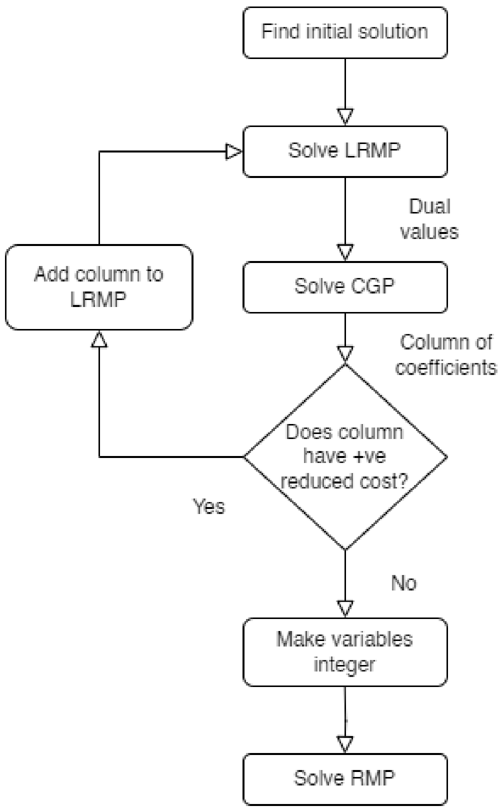
Algorithm The full column generation procedure for solving LPs is illustrated in the flow chart in Figure 3.3a. It relies upon having at least one initial column for which the MP has a feasible solution, this is typically heuristically computed. From here the RMP is solved using the initial column(s) and the dual values are extracted to model reduced cost as the objective to the CGP. The CGP is then solved to find the column with the greatest reduced cost. A check is then made to see whether the column is an improving one, i.e. it has a positive reduced cost if maximising. If so, it is added to the RMP and the process repeats, otherwise the solution to the RMP is optimal and the process terminates.

Termination Criteria It should be noted, that after any iteration of the procedure we have access to a measure of solution quality, If we denote $\boldsymbol{x}_G^*$, the optimal solution to the RMP for the variables generated so far, then $LB = \boldsymbol{c}_G^T \boldsymbol{x}_G^*$ is a lower bound on the optimal solution. Likewise, if we solve the CGP (3.10) to optimality and denote ρ^* as the optimal reduced cost, then if an upper bound on the variable values holds for the MP: $K \geq \sum_{i=1}^{n} x_i$, since it is not possible to increase the objective by more than $\rho^* K$, we also have an upper bound on each iteration: $UB = \boldsymbol{c}_G^T \boldsymbol{x}_G^* + \rho^* K$. Instead of checking whether the solution is optimal, we can instead compute the difference between the bounds ϵ, known as the optimality *gap*:

$$\epsilon = \frac{UB - LB}{UB} \tag{3.11}$$



(a) Procedure for solving an LP.



(b) Procedure for solving an MIP.

Figure 3.3: Flowchart showing the column generation procedure for LPs (a), versus MIPs (b). In (b), we solve a linear relaxation of the RMP (LRMP). When the column generation procedure terminates, the variables are made integer once more and the RMP is re-solved as a MIP.

If we define an acceptable tolerance on optimality, then we are free to terminate the procedure when the gap falls below a predetermined threshold. If this is the case, then the solution is known as an ϵ-*optimal* solution.

Column Generation for Mixed Integer Programs The algorithm defined in Figure 3.3a is only valid for continuous convex models (which includes LPs). As mentioned in Section 2.2.1, many interesting planning and scheduling problems feature variables whose values are *discrete* and are solved as a MIP. Such problems are non-continuous and non-convex. In order to solve MIPs using column generation, it is typical to relax the integer restriction on the variables, such that the relaxed RMP, referred to as the *Linear Restricted Master Problem* (LRMP), is an LP which can be solved using column generation. When the column generation procedure terminates, the integrality of variables are reintroduced and the RMP is solved as a MIP, as per Figure 3.3b. To guarantee a valid integer solution to the RMP, we must start the column generation procedure with a valid integer solution. This is typically generated using domain knowledge.

Solving a MIP using column generation represents adding columns at the root node of the branch and bound tree and therefore does not guarantee optimality. However, it is typically a good heuristic and its efficacy has been proven on many domains [132]. In order to guarantee optimality, one must employ the branch and price method [133]. In the branch and price method, each local node in the branch and bound tree is solved using column generation. Branch and price is outwith the scope of this thesis and so for interested parties we refer the reader to Barnhart et al. [133] for a comprehensive overview.

3.4 Example: Cutting Stock

The previous section focused on the theory behind why column generation works. However, the goal of this thesis is to show how column generation can be applied to develop new algorithms for solving practical problems. The decomposition into a column generation framework is not always clear. In this section, we show how the theory can be applied to solve practical optimization problems, with reference to the cutting stock problem.

Since its introduction by Kantorovic [134], the cutting stock problem has become a core example in many classical OR textbooks [131] and remains one of the most well studied examples of the application of column generation. The cutting stock problem has practical applications in a variety of industries, for example forestry [135, 136], carpentry [137], paper [138], construction [139] and steel [140].

In this problem, we are tasked with deciding how to cut pieces of stock of standard size, into orders of specified size while ensuring that we minimize the quantity of stock material used. Consider that we have a set of orders $\mathcal{O}$ and each order $o \in \mathcal{O}$ has a width w_o, and quantity q_o. The stock is a fixed width W_s.

3.4.1 Compact Model

A compact formulation to solve the cutting stock problem was introduced in Kantorovic [134]. In this formulation, we assume a known upper bound K, on the quantity of stock material used. We introduce binary variables $x_i \in \{0, 1\}$ for $i = 1, 2, \dots K$, which states whether stock i is cut. Likewise we enumerate a combinatorial number of integer variables $x_{i,o} \in \mathbb{Z}_+$ which define the number of times order o is satisfied in stock i. The full formulation is provided in (3.12):

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{K} x_i \\ \text{subject to} \quad & \sum_{i=1}^{K} x_{i,o} \geq q_o, && o \in \mathcal{O} \\ & \sum_{o \in \mathcal{O}} w_o x_{i,o} \leq W_s x_i, && i = 1, 2, \dots K \\ & x_i \in \{0, 1\}, && i = 1, 2, \dots K \\ & x_{i,o} \in \mathbb{Z}_+, && i = 1, 2, \dots, K, \ o \in \mathcal{O} \end{aligned} \tag{3.12}$$

The objective is to minimize the quantity of stock material cut. The first constraint enforces that the required quantity for each order is satisfied. The second constraint states that the total width of all orders cut from a given stock, cannot exceed the width of the stock material.

3.4.2 Column Generation Model

Gilmore and Gomory [16, 17] note that solving (3.12) is intractable for all but the smallest instances. Furthermore solving an LP relaxation of the compact model is not guaranteed to give a tight lower bound. Martello and Toth [141] prove that the lower bound can be as low as 50% of the integer objective value. This is a result of the fact that many fractional solutions (ways of cutting each stock) may not be feasible if integrality were to be reinforced, and so the optimal integer solution can be quite different. To counter this, Gilmore and Gomory [16] proposed enumerating all valid cutting patterns a priori. Each instance of stock can be cut into a set of patterns $\mathcal{P}$. If we denote $x_p \in \mathbb{Z}_+$, as the quantity of pattern $p \in \mathcal{P}$ used and $a_{p,o}$ as the number of times order o appears in pattern p, then we can write the MP as per:

$$\begin{aligned} \text{minimize} \quad & \sum_{p \in \mathcal{P}} x_p \\ \text{subject to} \quad & \sum_{p \in \mathcal{P}} a_{p,o} x_p \geq q_o, \quad o \in \mathcal{O}, \ \langle y_o \rangle \\ & x_p \in \mathbb{Z}_+, \quad p \in \mathcal{P} \end{aligned} \tag{3.13}$$

Now, the constraint in (3.13) simply states that each order quantity must be satisfied within the total patterns used to cut the stock. Note that there can be exponentially many patterns; Johnson et al. [142] comment that real world applications of the cutting stock problem in the paper industry can have billions of viable patterns. Consequently the number of x_p variables can be prohibitive. In addition, we may not explicitly know in advance all possible patterns, and therefore enumerating the columns associated with all the patterns may not be practical. Instead, we can start with a small subset and use column generation to iteratively add new improving patterns. With each constraint, we associate a dual variable y_o such that we can write the reduced cost of including a new pattern as:

$$\rho_p = 1 - \sum_{o \in \mathcal{O}} a_{p,o} y_o \tag{3.14}$$

Note that we still need to constrain the patterns according to the width of the stock material. Furthermore since $y_o \geq 0$, solving $\max_{\boldsymbol{a}_p}\{\sum_{o\in\mathcal{O}} a_{p,o}y_o\}$ is the same as solving $\min_{a_p}\{1 - \sum_{o\in\mathcal{O}} a_{p,o}y_o\}$. We can formalise the CGP as:

$$\begin{aligned} \text{maximize} \quad & \sum_{o \in \mathcal{O}} y_o \, a_{p,o} \\ \text{subject to} \quad & \sum_{o \in \mathcal{O}} w_o a_{p,o} \leq W_s \\ & \qquad a_{p,o} \in \mathbb{Z}_+ \quad o \in \mathcal{O} \end{aligned} \tag{3.15}$$

The CGP (3.15), is a knapsack problem, a well-studied problem in combinatorial optimization which seeks to select the best combination of items to include in a knapsack such that the value is maximized. Here, we wish to select the best combination of orders to include in the pattern. The dual value y_o, can be considered as the value of selecting one unit of order o in the pattern. Such problems are known to be NP-complete, however pseudo-polynomial time algorithms exist based on dynamic programming. In addition, it is not necessary to solve this sub problem to optimality, we only require an improving column. Efficient approximation schemes also exist capable of accelerating the procedure. Hence, in practice, (3.15) can be solved efficiently. Experimental validation is provided in numerous references, for example Vanderbeck [143].

3.4.3 Running Example

An example cutting stock problem is provided in Figure 3.4. When solving the cutting stock problem using column generation as per Figure 3.3b, we start with some initial solution. Coming up with a good initial solution is potentially chal-

Order Specification		
Order	**Width**	**Quantity**
1	w_1	5
2	w_2	1
3	w_3	3
4	w_4	3

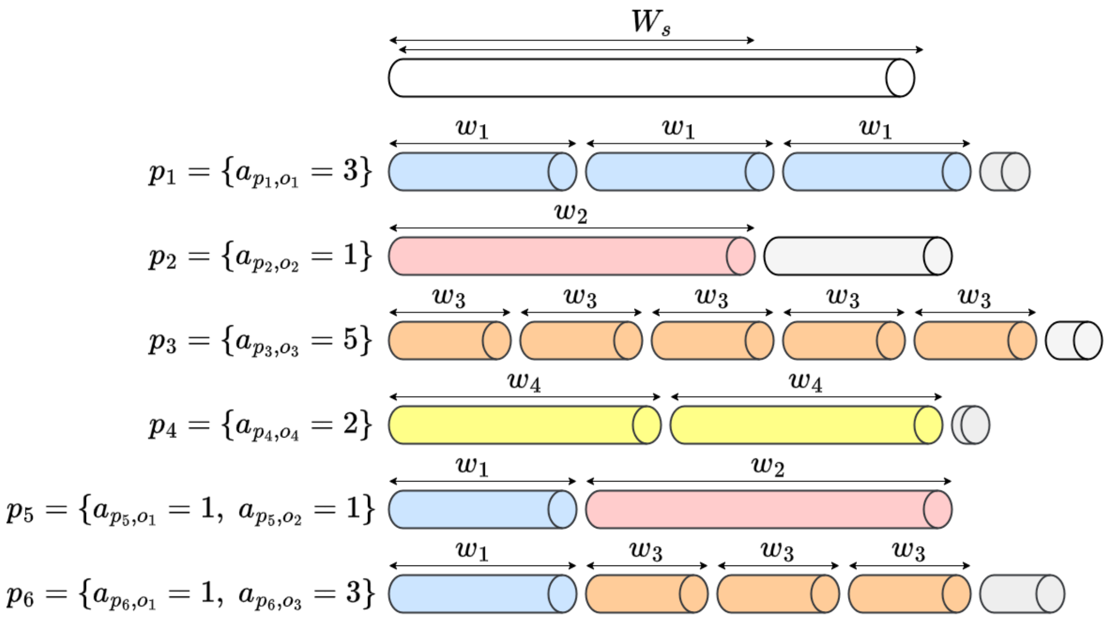

W_s
w_1
$p_1 = \{a_{p_1,o_1} = 3\}$
w_2
$p_2 = \{a_{p_2,o_2} = 1\}$
w_3
$p_3 = \{a_{p_3,o_3} = 5\}$
w_4
$p_4 = \{a_{p_4,o_4} = 2\}$
$p_5 = \{a_{p_5,o_1} = 1,\ a_{p_5,o_2} = 1\}$
$p_6 = \{a_{p_6,o_1} = 1,\ a_{p_6,o_3} = 3\}$

Figure 3.4: Figure showing an example cutting stock problem. The order quantities and widths are provided in the table. Each pattern is a feasible solution to the knapsack CGP (3.15). Among all feasible patterns, (3.15) is trying to find the best pattern for the given dual solution. One possible solution to this cutting stock problem would be to satisfy the orders using one instance of 3 distinct patterns: p_1, p_5, p_6, and 2 instances of the pattern p_4.

lenging, therefore it is common to kick start the procedure with a naive one. For the toy example provided in Figure 3.4, we could simply compute the maximum amount of each order we could include in the stock by taking the floor: $p_i = \lfloor W_s / w_i \rfloor$. This results in 4 initial patterns:

$$p_1 = \{a_{p_1,o_1} = 3\},\ p_2 = \{a_{p_2,o_2} = 1\},\ p_3 = \{a_{p_3,o_3} = 5\},\ p_4 = \{a_{p_4,o_4} = 2\}$$

With our initial patterns added, we would then solve a linear relaxation of (3.13). The solution to the LRMP would be to use 1.67, 1, 1 and 1.5 instances of patterns p_1, p_2, p_3 and p_4 respectively with a total cost of 5.17:

$$x_{p_1} = 1.67,\ x_{p_2} = 1,\ x_{p_3} = 1,\ x_{p_4} = 1.5$$

The dual variables would then be extracted using the appropriate function call from the LP solver of choice and used for the objective to the CGP (3.15). This could then be solved resulting in a pattern p_5, containing 1 instance of o_1 and 1 instance of o_2:

$$p_5 = \{a_{p_5,o_1} = 1,\ a_{p_5,o_2} = 1\}$$

We would then check that the reduced cost is negative for the given dual values y_{o_1}. y_{o_2} using (3.14): $1 - y_{o_1} - y_{o_2} < 0$. We find that it is, so the column is an improving one so we can add it to the LRMP.

With p_5 added, the LRMP would choose to select p_5 over p_2, since it contains an additional instance of o_1. The solution would therefore contain 1.33, 1, 1.5 and 1 instances of p_1, p_3, p_4 and p_5 with a total cost of 4.83:

$$x_{p_1} = 1.33,\ x_{p_3} = 1,\ x_{p_4} = 1.5,\ x_{p_5} = 1$$

The dual values would once more be extracted and the CGP solved resulting in pattern p_6 containing 1 instance of o_1 and 3 instances of o_3:

$$p_6 = \{a_{p_6,o_1} = 1,\ a_{p_6,o_3} = 3\}$$

Once more, we check that the reduced cost is negative for the given dual values y_{o_1}, y_{o_3}: $1 - y_{o_1} - 3y_{o_3} < 0$. The reduced cost is found to be negative and so the column is an improving one and so we add it to the LRMP.

We solve the LRMP once more and decide to use p_6 instead of p_3 such that our solution contains 1, 1.5, 1 and 1 instances of p_1, p_4, p_5 and p_6 with a total cost of 4.5:

$$x_{p_1} = 1,\ x_{p_4} = 1.5,\ x_{p_5} = 1,\ x_{p_6} = 1$$

Finally, we extract the dual values and find that there is no improving pattern, the optimal solution to (3.15) is less than or equal to 1 (the reduced cost (3.14) is non-negative). We would then solve (3.13) ensuring that the integrality of variables is reinforced, i.e. $x_p \in \mathbb{Z}_+$. Once integrality is reinforced, we note that taking 1.5 instances of p_4 (as per the LRMP) is not possible. The resulting solution to the RMP would require 1, 2, 1 and 1 instances of p_1, p_4, p_5 and p_6 respectively with a total cost of 5:

$$x_{p_1} = 1,\ x_{p_4} = 2,\ x_{p_5} = 1,\ x_{p_6} = 1$$

The optimal solution to the LRMP is a lower bound on the optimal solution, while the solution to the RMP is an upper bound. The resulting optimality gap can be computed as per:

$$\epsilon = \frac{5 - 4.5}{5} \times 100 = 10\%$$

Chapter 4

SLA Aware VNF Placement and Routing using Column Generation

"We are all now connected by the Internet, like neurons in a giant brain."

– Stephen Hawking

4.1 Introduction

In contrast to previous generations of mobile networks, the 5th Generation (5G) of networks will use network slicing, with multiple virtual network *slices*, overlaying a shared physical infrastructure [144]. Network slicing enables Internet Service Providers (ISP) to optimize infrastructure usage; while delivering tailored network services. Each *slice* corresponds to a particular service use case (e.g. industrial automation, autonomous driving and Ultra High Definition (UHD) video streaming [23]), each with contrasting and often competing requirements in terms of latency, throughput and availability [1]. These requirements are characterised by a set of Key Performance Indicators (KPI) quantified in the Service Level Agreement (SLA): a contract reached between the ISP and customer. Providing Quality of Service (QoS) means delivering the end-to-end service subject to the constraints imposed by the SLA.

Each slice is comprised of a number of Service Function Chains (SFC), where each SFC is a request to route and process traffic via an ordered sequence of Network Functions (NF) (e.g. load balancer, traffic monitor, firewall) [145]. Within the Network Function Virtualization (NFV) paradigm, NFs are Virtual Network Functions (VNF) implemented in software running on industry standard high volume servers [146]. Finding a suitable placement of the VNFs within the network, and subsequently routing the SFCs is a hard problem which has attracted significant attention and has been designated the VNF Placement and Routing Problem (VNF-PRP) [147].

Numerous papers have been presented in recent years tackling different variations of this problem. However, these approaches typically do one of the following: 1) minimize operational expenditure while neglecting QoS [148–150], 2) minimize specific QoS terms such as latency [151–154] or 3) model SLA constraints as hard constraints [147, 155]. When applied to network slicing, the prior may result in a VNF placement guaranteed to violate QoS for some slices, the second offers no distinction between the different QoS requirements of each network slice and the latter will simply result in no solution when it is not possible to satisfy the QoS constraints for a particular SFC.

In this chapter we present a VNF-PRP algorithm based on column generation, in which we iteratively solve a Restricted Master Problem (RMP), which optimizes the placement, replication and routing of the VNFs given the SFC paths generated so far; and a constrained shortest path Column Generation Problem (CGP) which generates new, improving paths. We treat SLA constraints (throughput, latency and availability) as soft constraints and then minimize SLA violation cost, meaning that we satisfy all SLA constraints *if possible* otherwise we satisfy *as many of them as possible.* Unlike many prior studies, our approach provides a full placement, replication and routing solution. In particular we compute: 1) the quantity of replicas of each required VNF, 2) a placement of each VNF onto compute nodes, 3) a number of paths for each SFC and 4) the fraction of flow to send down each path. We experimentally validated our approach on a realistic Mobile Edge Cloud (MEC) architecture generated using SNDlib benchmarks [156]. We show that for realistic sized instances (28 vertices, 41 edges, 700 SFCs), our approach is typically able to find near-optimal solutions within a practical time-frame.

The structure of this chapter is as follows. In Section 4.2 we introduce definitions related to the VNF-PRP. In Section 4.3 we review relevant literature and place the contribution of this chapter in context with respect to related work. In Section 4.4 we give a motivating example based on the MEC architecture to highlight the challenges associated with VNF-PRP. In Section 4.5 we show how the VNF-PRP can be decomposed into a column generation framework. In Sections 4.6 and 4.7 we describe the setup and results of our experimental validation. We conclude and address avenues for future research in Section 4.8.

4.2 Background

Network Slicing In recent years, there has been an emergence of novel, connected devices such as fitness-monitoring wearables, smart home appliances and energy monitoring devices [157]. These devices are capable of sensing, storing and processing large quantities of data and are collectively known as the Internet of Things (IoT). IoT promises to revolutionise the way we live our lives, from the development of smart cities and energy grids, to automated medical monitoring, factories and vehicles [158]. To make this possible, ISPs must decide how to host the services, each with their own diverse requirements, while making the most of the physical network infrastructure at their disposal.

Network slicing is the technology that makes this possible, in which multiple virtual network *slices* are overlaid on a shared physical infrastructure [144], thus optimizing infrastructure usage. Three general categories of network slices expected in 5G have been identified [159] and are summarised below, with the equivalent KPI requirements outlined in Figure 4.1:

1. **Enhanced Mobile Broadband (eMBB)** High speed mobile internet with uniform QoS constraints for human-centric use cases.

2. **Ultra Reliable Low Latency Communications (URLLC)** Communications for critical use-cases requiring extremely low latency and high reliability.

3. **Massive Machine Type Communications (mMTC)** Communications between a large number of connected IoT devices characterised by low-volume intermittent data transmission and high connection density.

Each slice has heterogeneous often competing requirements which makes slicing the network while maintaining QoS a challenging problem. These numeric requirements are defined according to industrial standards [23] in the SLA.

Network Function Virtualization Within each slice, there are a number of requests to access the service, denoted SFCs. Each SFC is a request to route and process traffic via an ordered sequence of NFs [145], where NFs are blocks within a physical network infrastructure which perform a well defined function, for example a firewall or load-balancer. NFs were traditionally built into specialised proprietary hardware known as middle-boxes. However, this hardware-centric approach incurred high capital and operational expenditure and was not scalable. Whenever a new service was to be provided, ISPs were required to purchase and

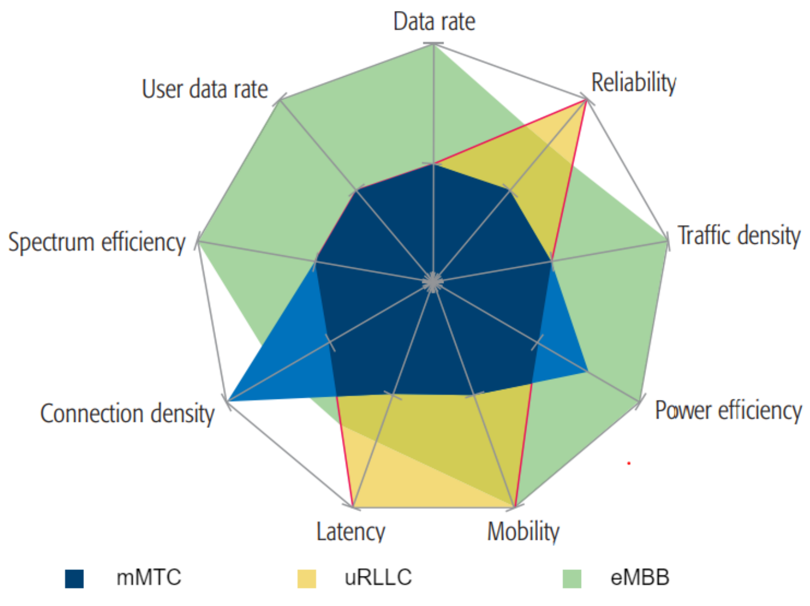


Figure 4.1: Figure showing varying KPI requirements for different service groups (adapted from [1]).

install new expensive middle-boxes, train skilled workers on the installation and operation of the middle-boxes as well as find physical space within the network infrastructure [160–162]. Furthermore the middle-boxes could not be reconfigured for new functionality meaning they quickly became redundant [163, 164]. Network Function Virtualization (NFV) is a paradigm that offers to solve these issues by replacing NFs running on middle box hardware, with Virtual Network Functions (VNFs) implemented in software running on Virtual Machines (VM) or containers hosted on industry standard high volume servers [146, 165, 166]. NFV offers solutions to many of the challenges of 5G. It will make it easier to provision new services, increase scalability and improve energy efficiency thus reducing costs [167].

Cloud and Edge Computing Working in conjunction with network slicing and NFV to enable 5G are the paradigms of cloud and edge computing. In cloud computing, virtual resources such as VNFs, are hosted on a large shared infrastructure such as a centralised DC and accessed by customers on a pay-per-use model [168]. Cloud computing lowers costs associated with provisioning new services since ISPs no longer need to build and maintain their own DCs [167]. On the other hand, hosting all services in a centralised DC is an issue for latency sensitive services, which are required to be hosted on nodes close to the user [169]. Edge computing provides a solution to this, by enabling computation and data-processing to be performed on nodes located at the edge of the network, thus preventing delays associated with transmitting the data to the cloud [170].

VNF Placement and Routing Problem While NS, NFV and edge computing are some of the key enablers of 5G, they also introduce additional challenges which must be overcome. One of the most important challenges is: how to provision the VNFs within the network and route the SFC requests, while satisfying the diverse QoS constraints imposed by each network slice. This problem has been denoted the VNF Placement and Routing Problem (VNF-PRP) and will be the focus of this chapter.

4.3 Related Work

There has been a plethora of literature in recent years addressing the more general case: the VNF Placement Problem (VNF-PP). For a comprehensive overview, we refer the reader to a relevant survey [171, 172]. In this section we address only those which are most relevant. Sun et al. [172] classify the VNF placement problem into 4 distinct sub problems: the *chaining problem* computes the VNFs and outputs a Virtual Network Function Forwarding Graph (VNF-FG) (possible ways of chaining the VNFs) based on demand; the *embedding problem* uses the VNF-FG as input and maps it to the physical links in the network subject to bandwidth resources; the *placement problem* allocates VNFs to the network subject to compute and network resources and the *routing problem* takes SFCs as input and routes traffic through the respective VNFs. Past literature has primarily involved: a) solving a combination of these sub problems, b) using an optimization method, c) for a particular use case, d) optimizing some metric. This is the way in which this section will be structured, where Section 4.3.1 addresses the different combinations of sub problems, Section 4.3.2 reviews different solution techniques, Section 4.3.3 discusses specific use cases and Section 4.3.4 examines different optimization metrics. Throughout, we emphasize how the work outlined in this chapter fits into this categorization.

4.3.1 VNF Placement in General

Early work in literature tackled the problem of placement, chaining and embedding but neglected the flow routing. Cohen et al. [173] formalised the VNF-PP problem by drawing comparisons to well known OR problems; the facility location problem and the general assignment problem. They present a near-optimal algorithm based on linearly relaxing an Integer Linear Program (ILP) to an LP, and then rounding the optimal solution so that it is integral. However, services are considered as single middle-box modules which are placed on the network while minimizing distance to the user. Chaining of VNFs and the bandwidth capacity of the links are not considered. To deal with the transition to NFV, Moens and de Turck [148] considered a hybrid scenario composed of a combination of middlebox and VNFs. By taking inspiration from virtual network embedding [174], they develop an ILP which places VNFs onto the physical infrastructure while minimizing servers used. While their work considers the bandwidth and latency of network links, they do not consider the chaining of VNFs which is a key feature of VNF placement. Mehraghdam [175] consider the chaining of VNFs and hence maintain the VNF ordering, however they solve the chaining and placement problems sequentially. First, they enumerate a set of VNF-FGs and then use a Mixed Integer Quadratically Constrained Program (MIQCP) to embed them in the network while optimizing numerous objectives.

The problem we address is better compared to the VNF-PRP introduced by Addis et al. [147]. In this problem, SFCs are directly encoded in the optimization as a flow routing problem, where we try to route traffic demands down any number of paths, while simultaneously placing VNFs directly on the paths subject to the ordering constraints. Addis et al. encode this problem as a multi-objective ILP minimizing both servers used and maximum network link utilization. They then present a matheuristic based on prioritisation of objectives (they split the optimization problem into a number of objectives which are solved sequentially, as per lexicographic optimization) to solve larger problem instances. However, they do not consider the replication of VNFs; in reality VNFs are often replicated for load balancing, fault tolerance and to cope with demand. Carpio et al. [176] introduced the VNF-PP with replication which takes this into account. However, their solution method involves three sequential optimization phases: first they enumerate a set of viable paths for each SFC, then they find the optimal placement of the VNFs on the enumerated paths and finally, they see if it is possible to improve the solution by introducing replicas. Each sub problem is then solved using a genetic algorithm, meaning it is impossible to measure the quality of the solution. As far as we are aware, our approach is the first algorithm capable of generating bounded optimal solutions considering placement, routing and replication.

In this chapter, we assume the network traffic to be static and deterministic, such that we can compute a static solution to the VNF-PRP. In practice, network traffic is dynamic and uncertain and therefore an online solution requires the capability of adjusting the VNF placement and routing according to demand. Such a problem is referred to as the dynamic VNF-PRP [177, 178]. Typically, ISPs will plan a baseline network configuration according to a pre-determined pattern of network traffic. This baseline configuration can then be adjusted online in an iterative manner based on learned experience. On that note, we stress that the solution presented in this chapter is not a dynamic one but would be used to determine a baseline configuration. A potential dynamic solution utilising the techniques from this chapter is outlined in Section 6.2 of this thesis.

4.3.2 Solution Methods

In terms of approach, VNF-PP has predominantly been solved using exact or heuristic methods. A number of authors present ILPs [5, 147–151, 175, 179–187] which can exactly compute the optimal solution, however they are incapable of solving problem instances of a practical size. To achieve scalability, heuristics of varying flavours are often presented. Many authors present greedy algorithms [151, 180, 181, 185, 187], Bari et al. [149] presents a dynamic programming based heuristic, Addis et al. [147] present a math-heuristic, Ghaznavi et al. [183] and Luizelli et al. [150] present a local and binary search based heuristic and Carpio et al. [176] use genetic algorithms. While computationally more efficient, such approaches do not offer guarantees on solution quality.

On the other hand, column generation can be used as a heuristic to solve practical sized problems while offering bounded optimality. Column generation has proven one of the most effective techniques for solving vehicle routing problems, to which the VNF-PRP contains many similarities. The vehicle routing problem was first formally defined by Dantzig and Ramser [188], as how to serve a set of geographically distributed customers using a fleet of vehicles. Balinski et al. [189] were the first to note that the vehicle routing problem can be formulated as a set-covering problem using a set of predetermined paths. Enumerating the set of valid paths is not especially trivial, furthermore the number of possible paths on a graph is exponential. Desrosiers et al. [18] were the first to use column generation to solve the problem of vehicle routing with time windows (each delivery must be made within a fixed time), considering only the paths that were needed. In the VNF-PRP we can consider the customers as the VNFs, and the vehicles as the SFCs, such that we want to find a set of routing paths to deliver packages (data traffic) to the customers (VNFs) beginning from a depot (the source node) and ending at another depot (the sink node). Just as each customer in the vehicle routing problem with time windows must be served within a set time, each SFC must traverse the required VNFs to provide the service within the latency time window. Often in vehicle routing problem, there is a capacity constraint on how many vehicles can traverse a road at a given time. This is analogous to the bandwidth constraints in the VNF-PRP; there is a limit on how much traffic we can route down any network link.

Liu et al. [178] use column generation and exploit it's any-time property to dynamically place VNFs to satisfy new SFC requests, however their column generation sub problem has an exponential runtime. Huin et al. [190] tackle the VNF-PP in static scenarios and show that the pricing problem can be formulated as a shortest path problem with polynomial time complexity. However, their approach differs from ours as they do not consider the QoS constraints such as latency and availability and assume that each SFC is mapped to exactly one path, with the routing demands down each path known a priori. Furthermore, they assume that the VNF instances can be fractionally split in terms of CPU and RAM. This is not the case in practice and can result in infeasible, non-integer assignments of the VNFs to the computational resources.

4.3.3 Use Case

VNF-PP for 5G has some interesting characteristics which render many of the general placement strategies insufficient. Cao et al. [191] study the problem of VNF Forwarding Graph (VNF-FG) design and embedding in 5G networks. A two-step method is proposed to generate the VNF-FGs according to the SFC requests, then 4 genetic algorithms are used to embed the graphs and place the

VNFs on the network while minimizing bandwidth consumption and maximum link utilization. Agarwal et al. [192] present a heuristic approach to solve the VNF-PP in 5G networks while minimizing latency. They note that solving the VNF-PP ensures that the minimum compute resources required for each VNF are available on the host node; but that additional resources can be provided to allow the VNF to process traffic more quickly. The allocation problem uses a queuing model to schedule the compute resources to individual VNFs hosted on the same node. Both Cao et al. [191] and Agarwal et al. [192] focus solely on the mobile core; Zhang et al. [2] highlight that in order to satisfy the ultra low latency requirement of some 5G services, placing VNFs at the edge is mandatory. They present an Adaptive Interference Aware (AIA) based heuristic which places VNFs on network slices wihin an edge-cloud architecture. They optimize throughput of accepted requests subject to slice-specific latency constraints, however they do not consider availability and solve the VNF-FG design and embedding problems sequentially (similar to Mehraghdam et al. [175]). Rather than explicitly modelling availability, Mohan and Gurusamy [193] introduce a resilient VNF-PP ILP for network slicing in which the objective is to minimize the number of SFCs affected given any node was to fail. As far as we are aware, our approach is the first VNF-PP solution capable of satisfying the throughput, latency and availability constraints required in 5G network slices.

4.3.4 Optimization Metric

Prior QoS sensitive approaches to VNF placement have typically handled sub sets of the required KPIs present in the SLA. While most studies consider throughput, Mehraghdam et al. [175] highlighted the importance of considering path latency and performed a Pareto set analysis to show the trade off between latency, number of servers used and link utilization. Following from this, numerous papers have considered latency as either the objective [151, 152] or as a constraint [147, 149, 150, 194, 195]. Rather than considering latency as a hard constraint which must be satisfied, Ben Jemaa et al. [182] treat it as a soft constraint for which violation incurs a cost. SLA violation cost is then minimized alongside link utilisation and server usage using the weighted sum method. Their approach however does not consider resource availability constraints (outlined in Section 4.5.1) and solves the problem using an exact MILP which is not scalable. They consider a small use case with one cloudlet and one cloud server and do not consider flow routing in the optimization.

More recently, some effort has been placed on incorporating availability in VNF placement algorithms. Hmaity et al. [184] present resilient strategies for safe-guarding against specific failures: node failure, link failure and a combination of both. However these strategies do not quantify availability and therefore are incompatible with the well-defined, numeric SLA requirements. Vizaretta et al. [155] were the first to explicitly model SFC availability as a constraint, using the product of individual VNF, node and link availability. However they model SLAs as hard constraints which simply returns no solution when it is not possible to satisfy all SLA constraints. Furthermore, they do not consider replication, in reality it is impossible to satisfy high availability constraints using only one SFC. Yala et al. [153] solve a weighted bi-objective optimization problem minimizing cost, while maximizing availability, where availability is derived using the probability that a node will fail. Similarly, Carpio et al. [154] model availability considering replications in a multi-objective framework. To avoid the non-linearity of the availability function, they choose to optimize a linear inverse penalty function which does not directly quantify availability. Both these approaches consider availability as the objective; in reality different slices have significantly different availability requirements which is why we chose to explicitly model availability as a constraint for each SFC.

4.4 Motivating Example

5G networks use an MEC architecture [196] with a combination of centralised core Data Centers (DC) and localised NFV enabled edge cloud servers. Core DCs have potentially unbounded compute and memory resources (they can be scaled up by adding new servers to cope with increased demand), however they can often be some distance from the user leading to high latency; while edge servers are resource constrained but tend to be placed close to the access points, thus lowering latency [2].

An example of expected 5G network slices is provided in Figure 4.2. Here we have three slices hosted on the shared MEC infrastructure. Each slice corresponds to a different use case with different QoS requirements. The autonomous driving slice falls under the category of URLLC characterised by ultra low latency and high availability. The UHD video streaming slice can be considered a eMBB use case and is characterised by high data-rates. Finally, the smart city slice is composed of a number of connected IoT devices and falls under the category of mMTC.

In order to satisfy the low latency constraint for the autonomous driving slice, it may be beneficial to host the VNFs at the edge. The slice also requires high availability; replicating the VNFs and hosting them on different servers increases the availability, since in the event of a node failure, traffic can still be processed by the replica. The total throughput required for the SFC can then be split down two paths, one visiting the master VNFs and one visiting the replicas. The UHD streaming slice has high data-rates meaning that a large proportion of the overall network bandwidth must be allocated. Likewise, the VNFs have traffic processing limits and so must be replicated to cope with demand. The throughput can be split down each replica as per the autonomous driving case. Finally the IoT based smart city slice has high connection density but is not as sensitive to QoS and therefore can be placed according to resource availability.

Satisfying all competing constraints for these diverse slices makes the optimization problem incredibly complex. Solving the VNF-PP while neglecting flow routing offers no guarantee that the latency QoS constraint will be satisfied. Likewise, simply placing and routing the VNFs without considering replication may result in a violation of either the throughput or availability QoS constraint. Considering any combination of these metrics as the objective does not distinguish between the different requirements for each slice. Consequently, prior VNF-PP algorithms are not viable for 5G network slicing. We now formally define the problem we are tackling.

Definition 6 (VNF-PRP)**.** *The VNF-PRP is a tuple: $\langle \mathcal{G}, \mathcal{S} \rangle$, where $\mathcal{G}$ is the network topology graph and $\mathcal{S}$ is the set of SFCs. The graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, has a set of vertices $\mathcal{V}$, representing physical locations in the network; and edges $(i, j) \in \mathcal{E}$ representing physical connections between the locations. The set of vertices can be classified into distinct subsets: the set of switches, $\mathcal{V}^s$ and the set of computational server nodes with NFV functionality, $\mathcal{V}^n$. Each node $n \in \mathcal{V}^n$, has compute and memory resources C_n, M_n and availability A_n, while each link (i, j) has an associated bandwidth capacity and latency, B_{ij} and L_{ij} respectively.*

The set $\mathcal{S}$ is the set of SFC requests which must be hosted within the network, where each SFC request $s \in \mathcal{S}$, is defined as a tuple $s = \langle \mathcal{F}^s, (v_0, v_d), \mathcal{R}^s \rangle$. The set $\mathcal{F}^s$, is the ordered set of required VNFs for the SFC, where each VNF $f \in \mathcal{F}^s$, has compute and memory requirements C^f and M^f, availability A^f, processing latency L^f and throughput capacity T^f. The pair (v_0, v_d), represents the source and sink destination of the SFC traffic, where v_0 is the source node and v_d is the destination. The set $\mathcal{R}^s$, is the set of numeric QoS requirements composed of $\langle T^s, L^s, A^s \rangle$, where T^s, L^s and A^s are the throughput, latency and availability requirements respectively. The VNF-PRP problem we seek to solve involves computing: 1) the quantity of replicas of each required VNF, 2) a placement of each

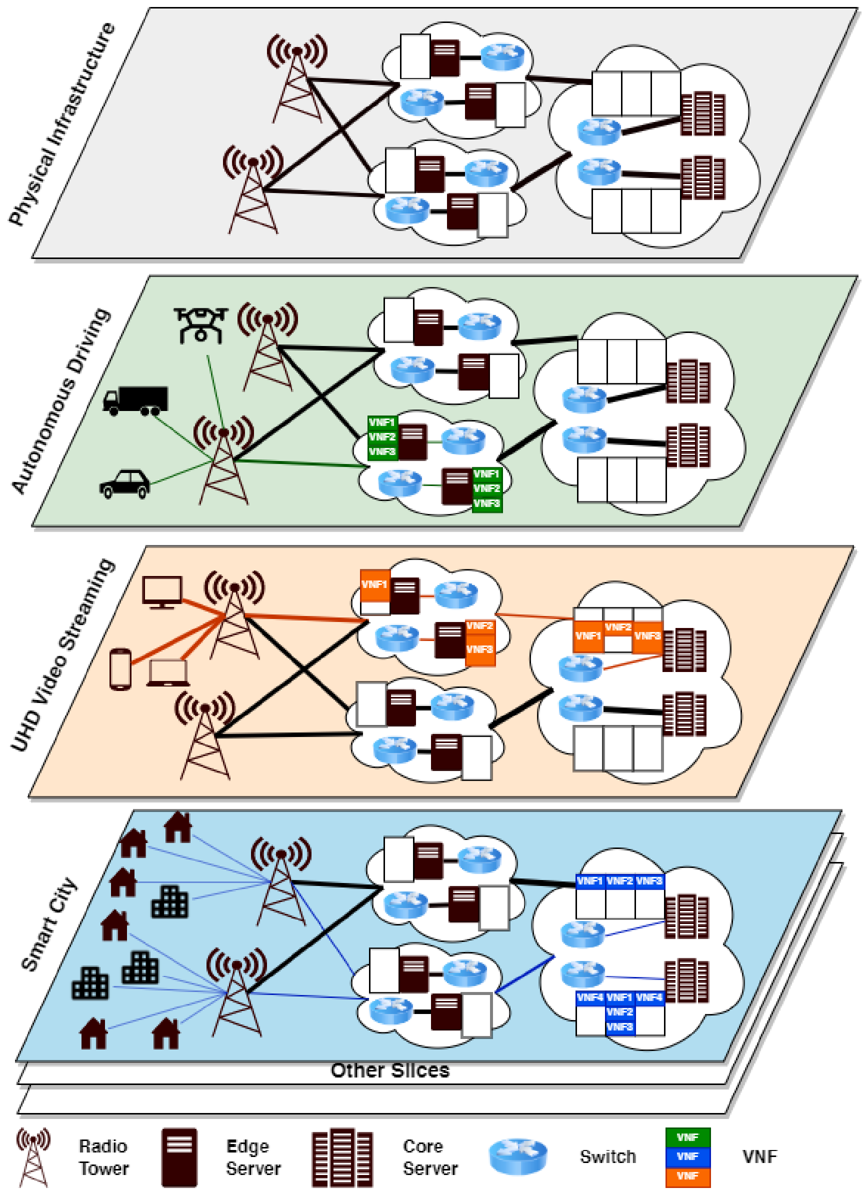


Figure 4.2: VNF placement example for 5G network slices (adapted from [2]). The VNFs required for the autonomous driving, UHD video streaming and smart city slices are shown in green, orange and blue respectively.

VNF onto compute nodes, 3) a number of paths for each SFC request and 4) the fraction of flow to send down each path.

4.5 Method

We solve the VNF-PRP via a column generation procedure as outlined in Algorithm 1, in which two optimization phases are iteratively solved:

1. The **Restricted Master Problem** (RMP) in line 7, which finds the number of replicas and placement of VNFs and the routing solution given the paths enumerated so far.

2. A **Column Generation Problem** (CGP) in line 10 for each SFC, in which we find the best new path to include in the RMP.

We will show in the coming section that each path is equivalent to a column in the RMP constraint matrix, henceforth we can use the terms *column* and *path* interchangeably. We initialise the set of paths $\mathcal{P}^s$, for each SFC using a heuristically generated valid path p in lines 1-3. We extract the dual values (Duals) from the solution to the LRMP in line 7, and use them to model the *reduced cost* which we set as the objective to the CGP. Since we are minimizing reduced cost, any path whose reduced cost is negative is called an *improving* column. If an improving column can be found (line 11), we add the new path to the set of paths enumerated for that SFC (line 13). The process then repeats (lines 6 - 16) until no improving columns can be found.

It should be noted that the RMP in this problem is a Mixed Integer Linear Program (MILP), however we solve a linear relaxation of this problem (LRMP) by replacing binary variables with continuous ones (as described in Section 3.3). This enables us to employ commercial solvers (e.g. Gurobi, Cplex) which do not permit adding variables at local nodes in the search tree. However, it does

Table 4.1: Master Problem Decision Variables

Sets	
$n \in \mathcal{V}$	The set of nodes in the network G.
$n \in \mathcal{V}^n$	The set of compute nodes in the network G.
$n \in \mathcal{V}^s$	The set of switch nodes in the network G.
$(i,j) \in \mathcal{E}$	The set of links in the network G.
$s \in \mathcal{S}$	The set of SFCs.
$f \in \mathcal{F}$	The set of VNFs.
Variables	
$x^p \in \mathbb{R}_+$	Fraction of flow through a path p for an SFC s.
$\phi_T^s \in \mathbb{R}_+$	Amount by which SFC s violates the SLA throughput.
$y_n^f \in \mathbb{Z}_+$	Number of instances of VNF f installed on node n.
$q_i^{s,f} \in \{0,1\}$	1 if i distinct nodes are hosting VNF f, for SFC s.
$\phi_A^s \in \{0,1\}$	1 if SFC s violates SLA availability, else 0.
$\beta_n^{s,f} \in \{0,1\}$	1 if VNF f for SFC s is hosted on node n, else 0.
Parameters	
W^s	Cost of violating SLA for SFC S.
C^f	CPU requirement for VNF f.
C_n	CPU resources of server node n.
M^f	Memory requirement for VNF f.
M_n	Memory resources of server node n.
T^s	Throughput requirement for SFC s.
T^f	Throughput capacity of each instance of VNF f.
B_{ij}	Bandwidth capacity of edge (i,j).
z_{ij}^p	Number of times an edge (i,j) occurs in a path p.
$\alpha_n^{p,f}$	Number of times a VNF f installed on node n processes traffic in path p.
$A_i^{s,f}$	Availability for i replicas of VNF f for SFC s.
A^s	Availability requirement for SFC s.
N	Min fraction of SFC flow required for a path to be considered in the availability calculation.
K_q^f	Max number of distinct nodes that can host a VNF f.
ϕ_L^p	1 if path p violates the SLA latency, else 0.

come with some caveats, for example it is only a heuristic. Once the column generation procedure has terminated, we restore the integrality of variables and solve the RMP as a MILP using the generated paths (line 17). Therefore, while the solution is not guaranteed to be optimal, it is integral.

Algorithm 1: Column Generation for VNF-PRP

```
   Input  : A network, $\mathcal{G}$
            A set of SFC requests, $\mathcal{S}$
   Output: Placement of VNFs and routing of SFC requests.
1  for $s$ in $\mathcal{S}$ do
2      $p$ := FindInitialPath($s$);
3      $\mathcal{P}^s$ := $\{p\}$;
4  end
5  Terminate := False;
6  while Terminate := False do
7      Cost, Duals := LRMP($\mathcal{G}$, $\mathcal{S}$);
8      Terminate := True;
9      for $s$ in $\mathcal{S}$ do
10         $p$ := CGP(Duals);
11         if $p.ReducedCost < 0$ then
12             Terminate := False;
13             $\mathcal{P}^s$.insert($p$);
14         end
15     end
16 end
17 Cost, Config := RMP($\mathcal{G}$, $\mathcal{S}$);
   Return: Cost, Config
```

4.5.1 Restricted Master Problem

If we were to enumerate the set containing all valid paths $\mathcal{P}^s$, for every SFC, then we can model the optimization problem as per Figure 5.8, with variables and parameters defined in Table 4.1. We refer to this as the Master Problem (MP).

$$\min_{x,y,\phi,\beta,q} \sum_{s\in\mathcal{S}} W^s \left(\phi_T^s + \phi_A^s + \sum_{p\in\mathcal{P}^s} \phi_L^p x^p \right)$$

$$\sum_{f\in\mathcal{F}} y_n^f C^f \leq C_n \qquad n \in \mathcal{V}^n \ (1)$$

$$\sum_{f\in\mathcal{F}} y_n^f M^f \leq M_n \qquad n \in \mathcal{V}^n \ (2)$$

$$\sum_{s\in\mathcal{S}} \sum_{p\in\mathcal{P}^s} z_{ij}^p T^s x^p \leq B_{ij} \qquad (i,j) \in \mathcal{E} \ \langle \mu_{ij} \rangle \ (3)$$

$$\sum_{p\in\mathcal{P}^s} x^p + \phi_T^s = 1 \qquad s \in \mathcal{S} \ \langle \pi^s \rangle \ (4)$$

$$\sum_{s\in\mathcal{S}} \sum_{p\in\mathcal{P}^s} \alpha_n^{p,f} T^s x^p \leq T^f y_n^f \qquad f \in \mathcal{F},\ n \in \mathcal{V}^n \langle \nu_n^f \rangle \ (5)$$

$$\sum_{f\in\mathcal{F}_s} \sum_{i=1}^{K_q^f} A_i^{s,f} q_i^{s,f} + \phi_A^s \geq \log A^s \qquad s \in \mathcal{S} \ (6)$$

$$\sum_{i=1}^{K_q^f} q_i^{s,f} = 1 \qquad s \in \mathcal{S},\ f \in \mathcal{F}^s \ (7)$$

$$\sum_{i=1}^{K_q^f} i q_i^{s,f} \leq \sum_{n\in\mathcal{N}} \beta_n^{s,f} \qquad s \in \mathcal{S},\ f \in \mathcal{F}^s \ (8)$$

$$\sum_{p\in\mathcal{P}^s} N \alpha_n^{p,f} x^p \geq \beta_n^{s,f} \qquad s \in \mathcal{S},\ f \in \mathcal{F}^s\ n \in \mathcal{V}^n \ \langle u_n^{s,f} \rangle \ (9)$$

Figure 4.3: Master Problem MILP

Objective ISPs must pay a fee if they fail to abide by the terms outlined in the SLA: denoted the SLA violation cost. As such, the SLAs are modelled as soft constraints for which violation incurs a cost (denoted W^s), the sum of which is minimized. If it is not possible to satisfy all SLAs, this approach should still satisfy the constraints *as best as possible.* It is trivial to extend this approach to minimize operational cost as a secondary objective. A weighted bi-objective framework could be utilised with the weights selected in accordance with the relative importance of minimizing operational costs and satisfying QoS. Nonetheless, the purpose of this chapter was to study the optimization of QoS, hence we leave this analysis as an avenue for future work.

Compute Constraints Constraints (1) and (2) ensure that the sum of CPU and memory requirements of the VNFs hosted on each node do not exceed the node resources.

Networking Constraints Constraint (3) says that the sum of all traffic flow passing through each edge must not exceed the bandwidth capacity of the edge. Constraint (4) ensures that as much of the required throughput for each SFC as possible must be routed down the paths. Constraint (5) says that the flow through all paths which consider a VNF to be installed on a particular node must be zero if that VNF is not installed on the node (i.e. $y_n^f = 0$). Conversely if $y_n^f > 0$, then the flow through all paths in the SFC which consider a VNF to be installed in that node can be at most $T^f y_n^f$.

It's worth mentioning that the ordering of the VNFs is explicitly encoded in the path (see Section 4.5.3). In order for a path to be selected, each sequential VNF must be installed on the required node encoded in the path through the $\alpha_n^{p,f}$ parameters: i.e. for $x^p > 0$, $y_n^f > 0$ if $\alpha_n^{p,f} = 1$. As such, Constraint (5) maintains the ordering of the VNFs encoded in the path.

Availability Constraints Availability of a network service refers to the total fraction of time that the service is functional and is typically computed using the Mean Time Between Failures (MTBF) and the Mean Time To Repair (MTTR):

$$A = \frac{MTBF}{MTBF + MTTR}$$

Here, the MTBF refers to the average time between failures occurring and the MTTR refers to the average time it takes to repair the failure. Consequently, the availability is the fraction of up-time (MTBF) over the total time including down-time (MTBF + MTTR).

As per prior studies [176,197,198], we consider the availability of an SFC A^s, as the probability that the SFC works, which is the probability that each sequential VNF works: $A^s = \Pi_{f \in \mathcal{F}} A_n^f$, where A_n^f is simply the product of the availability of the VNF, and the availability of the node that the VNF is hosted on: $A_n^f = A_n A^f$. The availability of the VNF is the probability that at least one of the replicas works, which is the complement of the probability that all replicas fail: $A^s = \Pi_{f \in \mathcal{F}^s} \left(1 - \Pi_{n \in \mathcal{N}} \left(1 - A_n^f\right)\right)$. A more accurate way to model SFC availability is by calculating the probability of at least one valid path being available as per Yang et al. [199]. This incorporates both node and link availability, however results in non-linear constraints which consequently renders the MP intractable. Since there is typically a rich path diversity between any nodes in modern networks [197], it should always be possible to reroute the traffic through the remaining instances of the VNFs in the event of simultaneous node failures. Hence, we are considering the availability of the nodes and VNFs (making sure that we have sufficient replicas of each of the VNFs running on the nodes) but not the availability of the links (we assume we can always reroute traffic between the VNFs running on the nodes). We assume that the availability of each server and VNF is the same, such that we can write the availability of the SFC as:

$$A^s = \Pi_{f \in \mathcal{F}^s} \left(1 - \left(1 - A_n^f\right)^{q^f}\right) \tag{4.1}$$

where q^f is the number of different nodes hosting a VNF. It's worth mentioning that we can have multiple replicas of a VNF running on one node. In which case, the availability $A_n^f = A_n \left(1 - (1 - A^f)^i\right)$, where i is the number of replicas running on node n. However, we note that $A_n \left(1 - (1 - A^f)^i\right) \geq A_n A^f$, and so Equation (4.1) is conservative. A similar justification can be made for assuming that the availability of all nodes/VNFs is the same.

Constraint (6) ensures that the SLA availability is satisfied and comes from Equation (4.1). If A^s is the required availability of the SFC, then we can rewrite Equation (4.1) as:

$$\sum_{f \in \mathcal{F}_s} \log \left(1 - \left(1 - A_n^f\right)^{q^f}\right) \geq \log A_s \tag{4.2}$$

where q^f is the number of nodes hosting VNF f. We make an assumption here that the number of different nodes hosting a VNF can be at most K_q^f. Assuming $A_n^f = 0.999$, even the availability requirement of 0.999999 can be achieved with $K_q^f = 3$, and so this is a reasonable assumption to make. We can then enumerate the function: $A_i^{s,f} = \log \left(1 - (1 - A_n^f)^i\right)$ for $i = 1, 2, .., K_q^f$ and introduce the binary variables $q_i^{s,f}$ which take a value of 1 if i nodes are hosting VNF f for an SFC s. If the LHS terms are not sufficient to satisfy the availability, then $\phi_A^s = 1$ and the SLA violation cost is incurred. Constraint (7) says that exactly one of these variables must have a value of 1 for each VNF. Constraint (8) forces the number of replicas to be equal to the number of distinct nodes that host the VNF. Constraint (9) forces the variable $\beta_n^{s,f}$ to take a value of 0 if the SFC flow through the node is less than $1/N$ of the total SFC flow. This forces the solution away from assigning arbitrarily small flows to paths in order to satisfy the availability.

For example if $N = 10$, then each path must have at least $0.1T^s$ flow through it. If one path is insufficient to satisfy availability, a valid solution may be to use two paths, one with $x^{p_1} = 0.9$ and one with $x^{p_2} = 0.1$.

4.5.2 Column Generation Problem

Note that there are exponentially many paths in the graph, and therefore enumerating all possible paths is computationally prohibitive. In addition, we are only interested in the paths from our base that are in the vicinity of the optimal solution. Instead, we iteratively solve a restricted version of the MP, that we refer to as the **Restricted Master Problem** (RMP), using a subset of the total paths. We can imagine that there are many other paths, whose variables x^p take a zero value in the RMP simplex basis and are therefore non-basic.

Each non-basic path variable x^p, at index k, in decision vector x has an associated *column* of coefficients A^k:

$$\boldsymbol{A}^k = \begin{pmatrix} \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{z}_{ij}^{p}\boldsymbol{T}^s & \boldsymbol{1} & \boldsymbol{\alpha}_n^{p,f}\boldsymbol{T}^s & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & -\boldsymbol{N}\boldsymbol{\alpha}_n^{p,f} \end{pmatrix}^T$$

Where each value in $\boldsymbol{A}^k$ is a vector of coefficients from constraints (1-9). Each path is directly encoded in the column, and is defined by precisely 2 variables: $\alpha_n^{p,f}$, which VNFs are installed on which nodes in the path and z_{ij}^p, how many times each edge occurs in the path.

The dual variables for each constraint are shown in Figure 4.3 enclosed by $\langle\rangle$. Since constraints (3, 4, 5 and 9) are the only constraints containing the path variable x^p (all other coefficients in $\boldsymbol{A}^k$ are 0), we denote $\boldsymbol{y} = \begin{pmatrix} \boldsymbol{\mu}_{ij} & \boldsymbol{\pi}^s & \boldsymbol{\nu}_n^f & \boldsymbol{u}_n^{s,f} \end{pmatrix}$, the dual vector. The reduced cost is given by:

$$\rho = W^s\phi_L^p - \pi^s - \sum_{(i,j)\in\mathcal{E}} T^s\mu_{ij}z_{ij}^p - \sum_{n\in\mathcal{V}^n}\sum_{f\in\mathcal{F}} \left(T^s\nu_n^f - Nu_n^{s,f}\right)\alpha_n^{p,f}$$

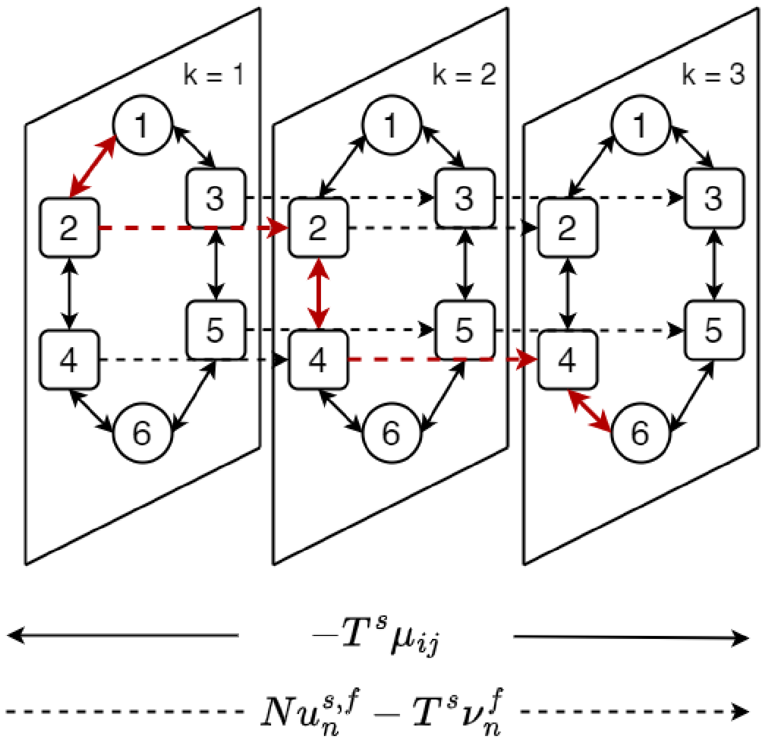


Figure 4.4: Diagram showing multi-layered graph.

The best new path is the one that minimizes the reduced cost: $\min_{\boldsymbol{z},\boldsymbol{\alpha},\phi}(\rho)$. For more details we refer the reader to Chapter 3.

4.5.3 Network Transformation

In this section, we show that the CGP can be solved as a constrained shortest path problem on a transformed network. For each SFC request, we construct an augmented network $\mathcal{G}^s = \langle \mathcal{V}^s, \mathcal{E}^s \rangle$. This network is a K-layered graph, where $K = |\mathcal{F}^s| + 1$, and each layer is a copy of the network $\mathcal{G}$. We introduce the binary variable z_{ij}^l, that says that an edge $(i, j) \in \mathcal{E}$, in layer l of the graph, is used in the path. From each node $n \in \mathcal{V}^n$ in layer l, we add an edge to the equivalent node in layer $l+1$. We have binary variables α_n^l that says that one of these edges from layer l is traversed. Traversing this edge represents an assignment of the VNF at index l of the set $\mathcal{F}^s$, to node n: i.e. if $\alpha_n^l = 1$, $\alpha_n^{p,f} = 1$. The problem

$$\min_{z,\alpha,\phi} W^s\phi^p_L - \pi^s + \sum_{(i,j)\in\mathcal{E}}\sum_{l=1}^{K}(-T^s\mu_{ij})z^l_{ij} + \sum_{n\in\mathcal{V}^n}\sum_{l=1}^{K-1}(Nu^{s,f}_n - T^s\nu^f_n)\alpha^l_n$$

$$\sum_{k:(j,k)\in\mathcal{G}} z^l_{jk} - \sum_{i:(i,j)\in\mathcal{G}} z^l_{ij} + \alpha^l_n - \alpha^{l-1}_n = 0 \qquad j,l : j^l \notin \{v^s_0, v^s_d\} \ (1)$$

$$\sum_{k:(j,k)\in\mathcal{G}} z^l_{jk} - \sum_{i:(i,j)\in\mathcal{G}} z^l_{ij} = 1 \qquad j,l : j^l = v^s_0 \ (2)$$

$$\sum_{k:(j,k)\in\mathcal{G}} z^l_{jk} - \sum_{i:(i,j)\in\mathcal{G}} z^l_{ij} = -1 \qquad j,l : j^l = v^s_d \ (3)$$

$$\sum_{(i,j)\in\mathcal{G}}\sum_{l=1}^{K} z^l_{ij}L_{ij} + \sum_{n\in\mathcal{V}^n}\sum_{l=1}^{K-1}\alpha^l_n L^l - M\phi^p_L \leq L^s \qquad (4)$$

Figure 4.5: Column Generation Problem ILP

is then to route flow from the SFC source node v_0 on layer 1 to the destination node v_d on layer K (referred to as v^s_0 and v^s_d respectively). An example is shown for a small network with 2 switches and 4 nodes in Figure 4.4. Consider an SFC request with source $v_0 = 1$, sink $v_d = 6$ and two VNFs. The sub problem would be to find a path from node 1 on layer 1, to node 6 on layer 3. A valid path: $p = \{(1^1, 2^1),\ (2^1, 2^2),\ (2^2, 4^2),\ (4^2, 4^3).\ (4^3, 6^3)\}$ is highlighted in red in Figure 4.4 and is equivalent to the column with non-zero terms: $z^p_{1,2} = 1$, $z^p_{2,4} = 1$, $z^p_{4,6} = 1$, $\alpha^{p,1}_2 = 1$ and $\alpha^{p,2}_4 = 1$. This transformation was first characterised by Huin et al. [190].

An ILP to solve this problem is provided in Figure 4.5. The objective is analogous to minimizing the reduced cost. Constraint (1) says the number of active edges into and out of each node that is neither the source or destination must be equal. Constraints (2) and (3) say that exactly one edge must be active out of and into the source and destination node respectively. Constraint (4) constrains the latency of the path based on the SLA. If the latency is not satisfied, the big M term corresponding to ϕ_L^p is activated and the SLA violation cost is incurred.

Note that the dual values are constants in this problem and the z_{ij}^l and α_n^l variables encode which links in Figure 4.4 are used. We can consider the terms $-T^s\mu_{ij}$ and $Nu_n^{s,f} - T^s\nu_n^f$ as the *edge weights*. If we were to remove constraint (4), along with the term $W^s\phi_L^p - \pi^s$ then what we are left with is just a shortest path problem. Since $\mu_{ij} \leq 0$, the edge weights $-T^s\mu_{ij}$ are all positive. Only the edges representing the assignment of a node to a VNF can be negative. Since these edges can only be traversed in one direction, the graph has no negative cycles. Such a problem can be solved efficiently using the Bellman-Ford algorithm [200] which has worst case time complexity $O(|V||E|)$, where $|V|$ and $|E|$ refer to the number of vertices and edges in the graph. In practice we found that employing commercial solvers to solve the ILP in Figure 4.5 was as fast as using Bellman-Ford and therefore that is the approach taken in this chapter. One could theoretically use Bellman-Ford to solve the CGP with the exception of constraint (4), check the reduced cost and if it results in an improving column then add the column and continue. If on the other hand, Bellman-Ford fails to result in an improving column, we could resort to solving the ILP from Figure 4.5.

4.6 Experimental Setup

As per prior studies [190], network topologies were taken from SNDlib [156]. Of the networks available, Abilene and Nobel EU were selected as they gave a broad range of network size. In order to simulate a realistic MEC architecture, a subset of nodes were selected to be core DCs, another subset were selected to be edge DCs and the remaining nodes were set as simple switches which can be access or destination points for SFC requests but have no NFV functionality. A summary of the graphs used is provided in Table 4.2.

Table 4.2: Networks Used

Name	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	**No. Core DC's**	**No. Edge DC's**
Abilene	12	15	2	6
Nobel EU	28	41	5	15

Since the edge DCs are more computationally constrained versus the larger core DCs, we set each edge DC as a single NFV node with 40 cores and 40GB of RAM. The Core DCs on the other hand were given 3 NFV nodes, each with 100 cores and 100GB of RAM. This was achieved by minor modification of the network; the node which was selected to be the core DC was set as the DC gateway switch, and was connected to three additional nodes. SFC requests could therefore access the gateway and then reach the required VNF on whichever node was hosting it. Apart from this, and due to the significant path diversity in the DCs and the fact that the internal DC latency is negligible, we did not model the internal DC topology. We note however, that a separate routing problem could be solved to route traffic between the DC gateways and the nodes if necessary. Using the coordinates of each node from SNDlib, we computed the length of each link as the distance between nodes, and then used this to compute the relative link latency. Latencies were then scaled in the range $[0, 2]$ in line with prior studies [2,201]. Bandwidth of each link was randomly sampled from $\{10, 40, 100\}$ Gbps.

The VNF data used is provided in Table 4.3 and was extracted from prior studies [2,195,202–204]. We assume an availability of 0.9999 and 0.999 for servers and VNFs respectively as per Wang et al. [205]. The network slice SFC requirements used are provided in Table 4.4 and have been extracted from prior studies [2,206], and industrial technical specifications [23,207].

The slice use cases were selected in line with the general categories identified in Section 4.2, where autonomous driving, UHD video streaming and smart city

Table 4.3: VNF requirements: C is the CPU requirement, M is the memory requirement, T is the throughput and L is the latency.

VNF	**C (cores)**	**M (GB)**	**T (Mbps)**	**L (ms)**
FW	4	4	600	0.8
TM	10	10	2000	0.1
IDS	8	8	600	0.01
NAT	16	16	3200	0.1
VOC	8	8	2320	0.25
ADNF	8	8	1500	0.1

FW: Firewall, TM: Traffic Monitor, IDS: Intrusion Detection System, NAT: Network Address Translator, VOC: Video Optimization Controller, ADNF: Autonomous Driving Network Function

correspond to URLLC, eMBB and mMTC respectively, The cost of violating each SFC was set according to the priority ranking from Jalalian et al. [206].

The number of SFC requests were varied (in the range 20 - 700), with each request being randomly assigned to slices from Table 4.4 and source and sink node. In order to simulate realistic traffic scenarios, the probability of sampling each slice type was estimated based on past internet traffic [208] (shown as Prob in Table 4.4). To scale the load on the network, we introduced a "load factor", which scaled the number of users accessing each SFC; for example a load factor 5 meant that every SFC in the network was being accessed by 5 users and therefore the total throughput would be $5T^s$.

Since there are no comparable models incorporating all the features considered in our approach, we compare our solutions to the LRMP LP solution after column generation. Note that the column generation procedure was solved to within 1% of optimality. Through using the reduced cost, we computed the lower bound from the LP solution (shown in the "LP Obj" column) which is a valid lower bound on the original MP. Similarly each RMP MILP was solved using branch and bound with an optimality gap of 1% and a time limit of 3600s. As per the

Table 4.4: SFC requirements for different slices: T is the throughput requirement, L is the latency, A is the availability, Prob refers to the probability of sampling the slice and W is the SLA violation cost.

Network Slice	**Required VNFs**	**T (Mbps)**	**L (ms)**	**A (%)**	**Prob**	W^s
Autonomous Driving	NAT-FW-TM-ADNF	10	5	99.999	0.1	3
UHD Streaming	NAT-FW-TM-VOC-IDS	200	100	-	0.4	2
Smart City	NAT-FW-IDS	0.1	-	-	0.5	1

LRMP, this still gives us a valid upper bound on the MP (shown in the "Obj" column). Gurobi was selected as the solver since it has been shown to be faster than similar competitors on recent benchmarks [209].[1]

4.7 Results

Experimental results are provided in Table 4.5.

SLA Violation Cost The total SLA violation cost is provided in the objective column "Obj." which shows the objective of the RMP MILP. Note that since we are minimizing, this is an upper bound on the optimal solution. Comparatively, the LP objective column "LP Obj." shows the optimal solution to the LRMP which is a lower bound on the optimal value. The difference between the upper and lower bound $(UB-LB)/UB$, is shown as a percentage in the "Gap" column.

Typically the model performs poorer on cases in which the number of SFCs and load factor is low. This is thought to be attributed to the high fractionality of the assignment variables y_n^f for these cases. In a number of instances in which the

[1]Source code and problem instances can be accessed online at: `https://anonymous.4open.science/r/VNFPP-CG-00DC`

Network	No. SFCs	Load Factor	LP Obj.	Obj.	Gap (%)	Total Pens.	Av. Pens.	Lt. Pens.	Tp. pens.	Run-time (s)	MILP Run-time (s)	CG Run-time (s)
abilene	20	1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.00	0.50
abilene	20	2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.49	0.00	0.49
abilene	20	3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.01	0.49
abilene	20	4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.62	0.01	0.61
abilene	20	5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.93	0.02	0.91
abilene	50	1	0.30	0.33	7.87	0.11	0.00	0.10	0.01	2.25	0.09	2.16
abilene	50	2	0.60	10.66	94.38	4.21	2.00	0.20	2.01	2.21	0.60	1.61
abilene	50	3	0.90	0.90	0.12	0.30	0.00	0.30	0.00	2.17	0.21	1.96
abilene	50	4	1.20	9.73	87.68	4.43	0.00	0.80	3.63	5.12	2.96	2.16
abilene	50	5	1.50	29.46	94.91	14.34	0.00	0.50	13.84	21.93	18.62	3.31
abilene	100	1	3.60	11.86	69.64	5.29	0.00	1.20	4.09	3.71	0.35	3.36
abilene	100	2	7.20	7.20	0.02	2.40	0.00	2.40	0.00	5.15	1.04	4.11
abilene	100	3	14.70	31.80	53.78	14.00	0.00	3.60	10.40	1034.45	1015.17	19.28
abilene	100	4	102.41	111.89	8.48	53.06	0.00	4.80	48.26	495.25	477.39	17.86
abilene	100	5	189.78	202.82	6.43	97.50	0.00	6.00	91.50	476.61	456.25	20.36
abilene	200	1	0.60	3.67	83.65	1.67	0.00	0.30	1.37	32.55	25.09	7.46
abilene	200	2	112.49	125.32	10.24	61.56	0.00	1.00	60.56	316.41	287.58	28.83
abilene	200	3	292.04	304.58	4.12	150.64	0.00	0.30	150.34	930.08	912.31	17.77
abilene	200	4	471.19	484.75	2.80	239.39	0.00	1.20	238.19	132.48	114.82	17.66
abilene	200	5	651.25	661.47	1.54	326.73	0.00	1.50	325.23	45.35	29.19	16.16
nobeleu	300	1	6.00	9.60	37.50	3.20	1.00	2.20	0.00	26.54	4.59	21.95
nobeleu	300	2	12.00	35.66	66.35	15.23	0.00	5.00	10.23	1154.69	1119.17	35.52
nobeleu	300	3	83.20	114.00	27.01	53.31	0.00	1.50	51.81	3783.57	3600.06	183.51
nobeleu	300	4	318.91	347.23	8.16	168.02	0.00	8.00	160.02	3757.26	3600.09	157.17
nobeleu	300	5	554.94	572.15	3.01	278.39	0.00	10.00	268.39	3812.27	3600.06	212.21
nobeleu	400	1	30.00	37.16	19.27	12.42	2.00	10.05	0.37	162.34	126.63	35.71
nobeleu	400	2	71.51	122.35	41.55	50.33	0.00	17.80	32.53	3682.93	3600.03	82.90
nobeleu	400	3	409.70	434.28	5.66	199.64	0.00	6.90	192.74	3778.47	3600.07	178.40
nobeleu	400	4	754.20	775.62	2.76	363.62	0.00	16.64	346.98	3755.44	3600.05	155.39
nobeleu	400	5	1090.90	1116.75	2.31	527.75	0.00	10.50	517.25	3756.83	3600.17	156.66
nobeleu	500	1	21.00	21.00	0.00	7.00	0.00	7.00	0.00	70.46	8.64	61.82
nobeleu	500	2	258.10	289.93	10.98	136.25	0.00	15.00	121.25	3913.12	3600.08	313.04
nobeleu	500	3	699.10	725.38	3.62	348.68	0.00	15.30	333.38	3867.85	3600.02	267.83
nobeleu	500	4	1127.55	1168.94	3.54	564.90	0.00	28.40	536.50	3871.82	3600.12	271.70
nobeleu	500	5	1570.07	1600.55	1.90	775.13	0.00	35.50	739.63	2056.64	1834.34	222.30
nobeleu	600	1	36.00	36.60	1.64	12.20	0.00	12.20	0.00	3674.62	3600.02	74.60
nobeleu	600	2	374.15	404.05	7.40	188.09	0.00	24.60	163.49	4010.19	3600.08	410.11
nobeleu	600	3	871.27	896.11	2.77	425.58	0.00	7.50	418.08	3918.77	3600.02	318.75
nobeleu	600	4	1366.79	1399.76	2.36	669.62	0.00	45.12	624.50	3828.99	3600.07	228.92
nobeleu	600	5	1862.69	1903.24	2.13	912.00	0.00	20.45	891.55	1928.65	1667.65	261.00
nobeleu	700	1	21.00	80.60	73.94	35.87	0.00	8.30	27.57	3720.72	3600.02	120.70
nobeleu	700	2	579.78	611.52	5.19	295.51	0.00	5.00	290.51	4178.42	3600.04	578.38
nobeleu	700	3	1170.59	1208.35	3.12	587.61	0.00	4.50	583.11	4058.58	3600.08	458.50
nobeleu	700	4	1774.12	1808.35	1.89	881.25	0.00	28.80	852.45	3981.58	3600.12	381.46
nobeleu	700	5	2370.00	2408.31	1.59	1174.83	0.00	7.00	1167.83	871.56	379.51	492.05

Table 4.5: Table showing experimental results.

network flow was low, the throughput constraint (constraint (5) in the LRMP) could be satisfied by setting $y_n^f \approx 0$. Paths were thus generated assuming this was a valid configuration. When integrality was restored and the variables y_n^f were forced to be integer, the ILP tried to set $y_n^f = 1$, but this resulted in a violation of constraints (1) and (2) for that particular node. Since the y_n^f variables were then required to be zero, and no paths had been generated for other configurations, these cases ended up paying some quantity of throughput penalties which were not paid in the LRMP. This is evident for example in Abilene with the number of SFCs 50 and load factor 4. Nonetheless, the more realistic sized cases tended to be solved to within 10% of optimality.

In the cases we ran, the model mostly managed to satisfy availability for the SFCs. For every autonomous driving SFC, the model will always choose to split the flow down at least two paths. For example the model could select one path with low latency to send 90% of the flow down, and then another with high latency to send 10% of the flow down. Since the latency penalty is fractional, this would contribute to a cost of $0.1W^s$. Conversely, since the availability penalty is binary, if we were to route all the flow down the low latency path, the model would pay a cost of W^s. Of course a different tradeoff could be achieved through a different parameterisation of the availability penalty versus latency.

Runtime The runtime for the procedure is highlighted in the column "Runtime". We also show a breakdown of the column generation runtime ("CG Runtime") versus the RMP MILP runtime ("MILP Runtime"). Note that the column generation runtime corresponds to the total runtime over the course of all CGP subproblems and LRMP LPs. It's worth noting that the column generation procedure is quite quick with all problems being solved within 600s. This is partially attributed to the fact that we solve the LRMP to within 1% of optimality, since we found that the column generation procedure struggles with convergence as the gap approaches zero. It was empirically found that 1% represented the best termination criteria, tightening the gap resulted in new columns being generated that did not result in a significant improvement to the objective. This was already deemed sufficient enough for practical purposes and so we did not investigate any stabilisation techniques. Most of the runtime is composed of solving the RMP MILP, since it involves solving a large MILP with many integer variables. We also found that branch and bound struggled to converge within a reasonable time for many instances when solving the RMP; hence our decision to also solve the RMP to within 1% of optimality.

Despite having significantly less variables than if we were to enumerate all paths, the RMP still has a significant number of variables (19595 continuous and 9623 integer for Nobel EU with 700 SFCs and load factor 5). As such solving the RMP MILP was still impractical for the larger instances. Instead, we set a time-limit of 1 hour (3600s) for Gurobi. When the problem could not be solved, the best incumbent solution was returned which still gives a valid upper bound on the optimal solution. Hence, where the MILP runtime is quoted as 3600s, the solver failed to find the optimal solution within the time-limit. Despite not satisfying the optimality termination criteria, the solver is typically able to find solutions that are within 10% of the optimal value. It's important to note that this gap is conservative since the LP lower bound is not tight, therefore the actual optimality gap may be much smaller than quoted in Table 4.5.

We would like to mention that we use integer variables to model the assignment of VNFs to compute nodes, y_n^f. Of course, this could be replaced with multiple binary variables $y_n^{f,i} \in \{0, 1\}$, where i refers to the number of instances of VNF f installed on node n. This could potentially improve the convergence efficiency of branch and bound in the RMP, however it was found that the model quickly reached reasonable incumbent solutions ($\pm 10\%$ within 100s) and so this comparison was left as an avenue for future work.

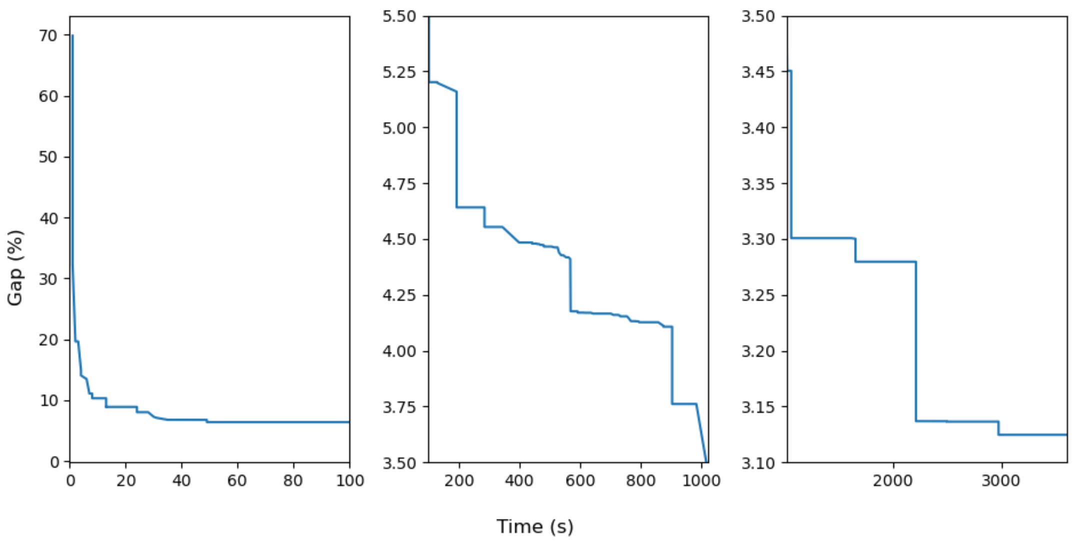


Figure 4.6: Plot showing gap versus MILP runtime for Nobel EU with 700 SFCs and load factor 3.

It's also worth pointing out that the RMP MILP tends to find a reasonable incumbent solution quite quickly. This is shown for the Nobel EU case with 700 SFCs and load factor 3 in Figure 4.6. After 100 seconds of solving, the optimality gap is just 5.20% with the remaining 3500s only resulting in a 2.08% reduction.

4.8 Conclusion

We present a QoS sensitive VNF-PRP algorithm which satisfies SLA constraints (latency, throughput and availability) *if possible*, otherwise it minimizes the cost of SLA violations. This allows us to generate solutions which are compliant with the diverse requirements of 5G network slices. We introduce one solution method using column generation, in which we iterate between generating *improving* paths and optimizing the placement and routing of the VNFs given the generated paths.

The column generation sub problem can be solved efficiently meaning that practical sized problems can be solved in a reasonable time. We experimentally validated our approach on a realistic MEC architecture generated using SNDlib benchmarks. We show that for realistic sized instances (28 vertices, 41 edges, 700 SFCs), our approach is typically able to find near-optimal solutions within a practical time-frame (±10% of optimal value within 1 hour).

We conclude by addressing some caveats and avenues for future work. First, we found that when the number of SFCs and the network load is low (low number of users/SFCs), the solution can be quite poor (gap of 94.91% for the worst case). This is a result of the fractionality of the VNF assignment variables as explained in Section 4.7. One solution would be to employ a branch and price framework [133] to add columns at local nodes in the search tree. This would permit finding the optimal solution but would require using specialised solvers (e.g. SCIP [210]). Implementation of a branch and price solution using SCIP would be a significantly involved engineering project requiring writing custom branching rules. While a Python interface for SCIP exists (see PySCIPOpt [211]), support is limited and column generation was providing good solutions, hence our decision not to pursue this avenue. Another potentially interesting avenue is a more comprehensive study of the trade-off between satisfying QoS and minimizing operational cost. Operational cost is typically a function of the bandwidth and number of nodes used. One could model the problem as a multi-objective optimization problem as explained in Section 4.5.1. By varying the weights, a Pareto front of solutions could be generated.

Chapter 5

A Column Generation Approach to Correlated Simple Temporal Networks

"A plan is what; a schedule is when. It takes both a plan and a schedule to get things done."

– Peter Turla

5.1 Introduction

Automated planning is the problem of selecting a sequence of actions which, when applied to some initial state, satisfies a number of goal conditions [30]. Temporal planning is a particular class of planning problems in which actions have durations. Once the plan has been found there remains the problem of executing the actions within the real world. Actions have a number of conditions which must be satisfied for their success to be guaranteed, therefore maintaining the correct order of the actions is crucial. This is increasingly difficult in temporal planning where external uncertainty factors can result in action durations taking significantly longer or shorter than those assumed in the planner. As such it becomes necessary to decide *when* to perform the actions such that this ordering is maintained. Such a problem falls within the realm of *scheduling*.

Simple Temporal Networks with Uncertainty (STNU) [212], are graphs used to represent and reason over scheduling problems involving uncertain durations. In an STNU, actions are denoted as time-point vertices on the graph, whereas the edges represent constraints on the duration between the time-points. A solution to an STNU, is a schedule at which to execute the time-points, such that the temporal constraints are satisfied. STNUs capture uncertainty in the problem through the inclusion of set-bounded *contingent links*. *Continuous probability distributions* are a more accurate representation of duration uncertainty; Probabilistic Simple Temporal Networks (PSTN) model the uncertain duration with a probability density function [13, 100].

When dealing with uncertainty in temporal networks, it is typical to classify the problem in terms of controllability [213], which states how sophisticated an execution strategy is allowed. Strong Controllability (SC) asks if there is a single schedule robust to all uncertain outcomes, i.e. all constraints are satisfied no matter what happens. However, PSTNs are rarely strongly controllable as the continuous probability distributions are unbounded.

A PSTN can be reduced to a strongly controllable STNU through *truncating* the distributions over durations. However, this discards some of the probability mass of the distribution, thus reducing the robustness of the schedule. A variety of approaches have been introduced for solving SC of PSTNs while maximizing robustness [13, 100–102]. However, these approaches either bound above the risk using Boole's inequality which offers no guarantee on optimizing robustness; or solve a generic non-linear optimization problem which can be computationally expensive. Furthermore, all prior approaches assume independence of uncontrollable outcomes, which does not always hold. As an example consider a network of drones to be used in the delivery of medical supplies to rural communities [214]. The drone must fly between locations, picking up and dropping off supplies before they expire. Each leg of the journey is subject to correlated temporal uncertainty as a result of weather factors such as wind speed and direction. In vehicle routing problems correlation has been shown to exist in travel times [215], with coefficients as high as 0.75 [216].

In this chapter we introduce the Correlated Simple Temporal Network (Corr-STN) which generalises the PSTN by removing the assumption of independence. We show that the problem of optimizing robustness is convex for a wide range of log-concave distributions. This allows us to solve Corr-STN SC using one of the many available convex optimization algorithms. We introduce one such approach leveraging the column generation method [217], in which we iteratively refine and optimize on an approximation of the distributions. This approach provides the decision maker the choice to trade-off numerical time spent with an acceptable optimality guarantee.

We test our approach on a number of drone delivery temporal planning problems. Corr-STNs are generated from the solution to the planning problem (the plan) and solved to compute a schedule maximizing robustness. We compare results against a linear program (LP) using Boole's inequality [101] and our approach assuming independence. We then perform Monte-Carlo simulations, simulating the execution of each schedule on the Corr-STN and compare the robustness (the total probability of success) and accuracy of solutions. We show that considering the correlations offers more robust schedules than using Boole's inequality or assuming independence. Although the robustness benefit of considering correlations varied substantially, we typically experienced greater improvements when the optimal robustness was low. When the optimal robustness was less than 0.5, considering correlation offered an average robustness improvement of 8.51% over using Boole's, and 3.50% over assuming independence. We also highlight that while Boole's is a bounding approximation of the true robustness, it can be grossly inaccurate, whereas assuming independence can be more accurate but is not guaranteed to give a conservative estimate of robustness. On the other hand, considering the correlation gives an accurate approximation of the true robustness but is more computationally expensive versus the other two approaches.

In Section 5.2 we introduce definitions related to STNUs, PSTNs, controllability and robustness. In Section 5.3 we review relevant literature and place the contribution of this chapter in context with respect to related work on PSTN SC. In Section 5.4 we motivate the importance of considering correlations through reference to a toy example and formally define the Corr-STN. In Section 5.5 we highlight how Corr-STN SC can be encoded as a convex optimization problem. In Section 5.6 we present one possible solution approach utilising the column generation method. In Sections 5.7 and 5.8 we describe the setup and results of our experimental evaluation. We conclude and address avenues for future research in Section 5.9.

5.2 Background

Simple Temporal Networks with Uncertainty Simple Temporal Networks (STN), outlined in Section 2.2.3, are graphs used to model scheduling problems involving actions. One of the key issues with STNs is the assumption that all time-points are *controllable*; we can control the duration that actions will take. STNUs [212] were introduced to model the temporal uncertainty inherent in the real world. In STNU semantics, a distinction is made between *contingent links*, for which the duration of the interval is uncertain, and *requirement links*, for which we can choose the duration.

Definition 7 (STNU). *A STNU is a tuple, $S^U = \langle T_c, T_u, C, G\rangle$ where $b \in T_c$ is*

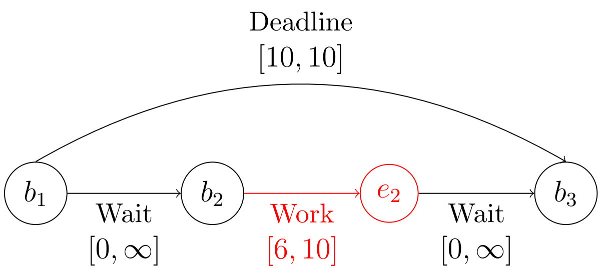


Figure 5.1: Example Simple Temporal Network with Uncertainty.

the set of controllable time-points and $e \in T_u$ is the set of un-controllable time-points, such that $t \in \{T_c \cup T_u\}$. The set C is the set of temporal requirement constraints between two time-points, normally written in the form $c(t_j, t_i) = t_j - t_i \in [l_{c,ij}, u_{c,ij}]$. The set G is the set of contingent links given in the form $g(e_i, b_i) = e_i - b_i \in [l_{g,i}, u_{g,i}]$. Here $l_{c \vee g}, u_{c \vee g}$ is the lower and upper limits for the constraint or contingent link respectively. Let $s\,(b) \in \mathbb{R}^+$ be the assignment of a value to the controllable time-point b. Let $o\,(e) \in \mathbb{R}^+$ be the value observed by an uncontrollable time-point e. A projection of a contingent link g_i is $\omega_i := o(e_i) - s(b_i)$.

Returning to the example introduced in Section 2.2.3, we assume that the student cannot choose exactly how long it will take for him to complete his assignment, but that it will take some duration between 6 and 10 days. An equivalent STNU is provided in Figure 5.1 and the relevant time points are listed below:

b_1 : Time at which the professor gives the project to the student.

b_2 : Time at which the student begins work on the project.

e_2 : Time at which the student finishes work on the project.

b_3 : Time of the deadline (When the student must finish the work by).

Note that the time-point e_2, is uncontrollable. The student is free to choose when to begin the project, but since the duration that it will take to perform the work is uncertain (denoted by contingent link $g(e_2, b_2)$ highlighted in red in Figure 5.1), the student cannot schedule the time that he will finish the work. An effective strategy must consider all possible projections of the contingent link.

Strong Controllability The concept of consistency was extended for STNUs by Vidal [213] to the notion of controllability. Split into three categories (strong, dynamic and weak) controllability can be considered as a way of classifying how much control is needed to satisfy all constraints. Strong controllability (SC) says that the STNU is consistent, regardless of the outcome of the uncertain contingent links.

Definition 8 (Strong Controllability)**.** *Denote Ω, the space of outcomes of the contingent links: $\Omega = \times_{g \in G}[l_g, u_g]$. Let the schedule s be the assignment: $s(b)$, $\forall b \in T_c$. An STNU S is said to be strongly controllable if: $\exists s \mid c(t_j, t_i) \in [l_{c,ij}, u_{c,ij}]$, $\forall c \in C$, $\forall \omega \in \Omega$.*

SC is a highly desirable property, as it offers the advantage that a fixed-time schedule can be computed offline which will work regardless of how the contingent links are realised at execution.

Without loss of generality, the requirement constraints can be separated into the set of controllable constraints C_c in the form: $c(b_j, b_i) = b_j - b_i \in [l_{c,ij}, u_{c,ij}]$ and uncontrollable constraints C_u in the form: $c(e_j, b_i) = e_j - b_i \in [l_{c,ij}, u_{c,ij}]$ (as per $c(e_2, b_1)$ in Section 5.4) or $c(b_j, e_i) = b_j - e_i \in [l_{c,ij}, u_{c,ij}]$ (as per $c(b_2, e_1)$ in Section 5.4). By substituting $e_j = b_j + \omega_j$ or $e_i = b_i + \omega_i$, we can write the uncontrollable constraints in terms of the controllable time-points, and the projection, as per Equations (5.1) and (5.2):

$$c(e_j, b_i) = b_j + \omega_j - b_i \in [l_{c,ij}, u_{c,ij}] \tag{5.1}$$

$$c(b_j, e_i) = b_j - b_i - \omega_i \in [l_{c,ij}, u_{c,ij}] \tag{5.2}$$

To check whether an STNU S is strongly controllable, it is sufficient to check that the requirement constraints are satisfied for the worst possible projection of the contingent links. This can be achieved by rearranging for ω, giving us the maximum and minimum value (u_g and l_g respectively) that the projection can take, while still satisfying the uncontrollable constraint. For constraints of the form present in Equation (5.1), the worst cases are provided in Equations (5.3) and (5.4):

$$\max_{\omega_j \in [l_{g,j}, u_{g,j}]} c(e_j, b_i) := u_{g,j} \leq b_i - b_j + u_{c,ij} \tag{5.3}$$

$$\min_{\omega_j \in [l_{g,j}, u_{g,j}]} c(e_j, b_i) := l_{g,j} \geq b_i - b_j + l_{c,ij} \tag{5.4}$$

And for constraints of the form present in Equation (5.2), the equivalent worst cases are presented in Equations (5.5) and (5.6):

$$\max_{\omega_i \in [l_{g,i}, u_{g,i}]} c(b_j, e_i) := u_{g,i} \leq b_j - b_i - l_{c,ij} \tag{5.5}$$

$$\min_{\omega_i \in [l_{g,i}, u_{g,i}]} c(b_j, e_i) := l_{g,i} \geq b_j - b_i - u_{c,ij} \tag{5.6}$$

For further details on solving STNU SC, we refer the reader to the work of Morris and Muscettola [218] or Cimatti et al. [219].

Returning to the STNU example from Figure 5.1, the schedule $s = \{b_1 := 0, b_2 := 0, b_3 := 10\}$ is a valid schedule for every projection $\omega_2 \in [6, 10]$ and so the STNU is SC.

Probabilistic Simple Temporal Networks When sufficient data is available, it is more accurate to model the space of projections of a contingent link by a probability density function. This allows the scheduling process to focus on the durations *most likely* to be realised at execution. Probabilistic Simple Temporal Networks (PSTN) were introduced by Tsamardinos et al. [100].

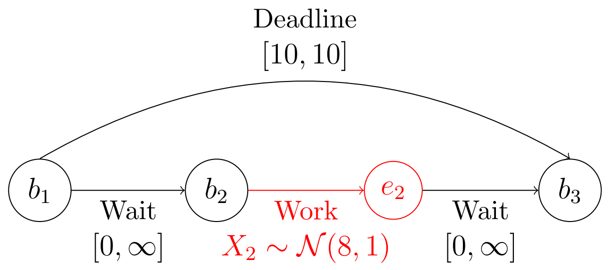


Figure 5.2: Example Probabilistic Simple Temporal Network.

Definition 9 (PSTN). *A PSTN is a tuple, $S^P = \langle T_c, T_u, C, D \rangle$, where T_c, T_u and C are as per the STNU. The set of probabilistic constraints D, are in the form $d(e_i, b_i) = e_i - b_i = X_i$, where X_i is a random variable with a set of outcomes Ω_i, probability density function $f(\omega_i)$ and cumulative probability function $F(z) = P(X_i \leq z)$.*

The STNU in Figure 5.1, offers no insight into the likelihood of the student taking 6 days versus 10 days to complete the assignment. Furthermore, it assumes that there is zero probability that the task will take longer than 10 days, or less than 6 days. This is not necessarily the case in practice. A more accurate representation may be to use a Gaussian distribution: the length of time that the student will take to perform the assignment can be described with a mean of 8 days, and a standard deviation of 1 day. An example PSTN is given in Figure 5.2.

PSTN SC and Risk It should be noted that the PSTN in Figure 5.2 is not SC as the Gaussian distribution is unbounded. Even if the student was to begin working on the assignment as soon as it was provided, there is a non-zero probability that the assignment will take significantly longer than 10 days. The probability of such an event occurring however is very small, and so considering these unlikely outcomes could be deemed overly conservative.

As a result, it is typical to truncate the distribution by neglecting the extreme, unlikely outcomes in the tails, i.e: $\Omega_{*i} = [l_{d,i}, u_{d,i}]$. If we denote by $d_*(e_i, b_i)$ the transformed probabilistic constraint with value defined by random variable X_{*i} and set of outcomes Ω_{*i}, then performing this transformation transforms the probabilistic constraint to a contingent link: $d_*(e_i, b_i) = e_i - b_i = X_{*i} \in [l_{d,i}, u_{d,i}] \equiv g(e_i, b_i)$. Applying this transformation to all $d \in D$, is equivalent to transforming the PSTN S^P, to an equivalent STNU S^{*U}. However the schedule is now only robust to the outcomes considered in S^{*U}. The probability mass excluded by performing this transformation is the *risk* of S^{*U}, while the probability mass considered is the *robustness*. This is shown for one particular probabilistic constraint in Figure 5.3, where the hatched area refers to the risk and the area between the dotted and dashed line is the robustness. We refer to Fang et al. [13] for a definition of robustness and risk, written in its equivalent form below.

Definition 10 (Robustness and Risk)**.** *Let s be a schedule and denote $c(s, \omega)$, the value of constraint $c \in C$, given the schedule s and outcome $\omega \in \Omega$. If for every $c \in C$, $c(s, \omega) \in [l_{c,ij}, u_{c,ij}]$: then $\omega \in \Omega_R \subseteq \Omega$. The robustness Γ, of a schedule s, is $P(\Omega_R)$, while the risk Δ, is $P(\bar{\Omega}_R)$, where $\bar{\Omega}_R$ denotes the complement of the set Ω_R.*

Since the joint probability functions, $P(\Omega_R)$ and $P(\bar{\Omega}_R)$ may be non-trivial, it is typical to treat each probabilistic constraint independently, such that the robustness can be evaluated: $\Gamma = \prod_{d \in D} P(l_d \leq X \leq u_d)$ and the risk: $\Delta = 1 - \prod_{d \in D} P(l_d \leq X \leq u_d)$. The values of u_d and l_d are determined through the SC relationships, Equations (5.3) to (5.6), by substituting l_g, u_g for l_d, u_d. To permit a linear formulation, it is possible to bound above the risk using Boole's inequality. Boole's inequality states that the probability of at least one event happening is strictly less than sum of the probabilities of the individual events. The risk can therefore be approximated as $\Delta = \sum_{d \in D} (1 - F(u_d) + F(l_d))$, while the robustness: $\Gamma = \sum_{d \in D} (F(u_d) - F(l_d))$.

Returning to the example PSTN from Figure 5.2, from Equations (5.3) to (5.6),

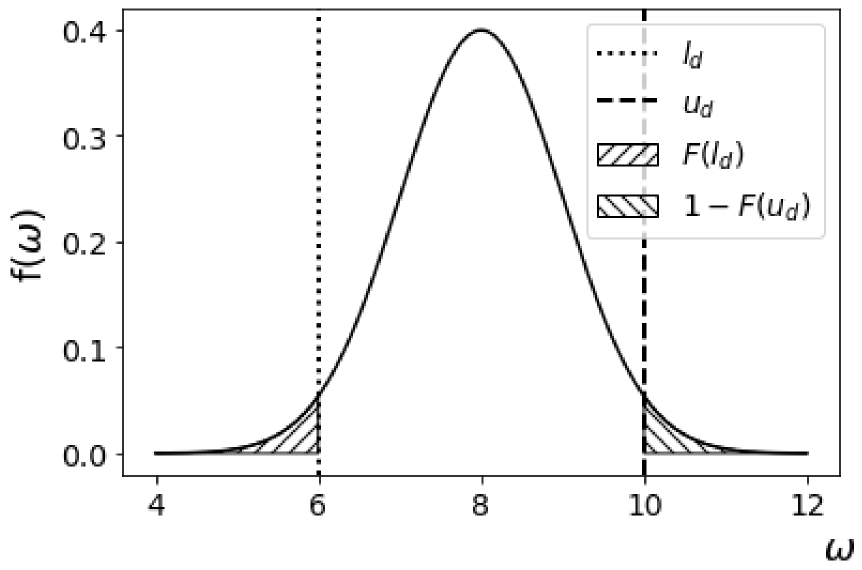


Figure 5.3: Figure showing risk associated with squeezing a probabilistic constraint to an equivalent contingent link.

to enforce SC we need the following conditions to hold:

$$b_2 - b_1 \leq \infty$$

$$b_2 - b_1 \geq 0$$

$$b_3 - b_1 = 10$$

$$u_{d,2} \leq b_3 - b_2$$

$$l_{d,2} \geq b_3 - b_2 - \infty$$

Assuming the schedule $s = \{b_1 := 0, b_2 := 0, b_3 := 10\}$, for SC to hold, we have that $u_{d,2} \leq 10$ and $l_{d,2} \geq -\infty$. We can transform the probabilistic link $d(e_2, b_2)$ to a contingent link $g(e_2, b_2) = e_2 - b_2 \in [-\infty, 10]$. The resulting STNU is SC with schedule s, robustness $\Gamma = P(-\infty \leq X_2 \leq 10)$ and risk $\Delta = 1 - P(-\infty \leq X_2 \leq 10)$.

5.3 Related work

5.3.1 Algorithms for PSTN SC

PSTNs were first introduced by Tsamardinos [100]. They motivated the need for such a framework by highlighting that prior STNU SC approaches [220], assume that the probability of durations existing outwith the bounds of contingent links is zero: which is not the case in practice. Instead, they explicitly compute the probability mass excluded by transforming a PSTN to a strongly controllable STNU. They show that robust execution of PSTNs should focus on finding the schedule that maximizes the probability of success (robustness). By assuming independence, they leverage Sequential Quadratic Programming to find static schedules: thus first solving the problem of PSTN SC.

Fang et al. [13] noted that maximizing robustness can lead to conservative solutions. To counter this, they introduced the Chance Constrained PSTN (CC-PSTN), by enforcing an allowable tolerance on the risk as a constraint in the system. Some other objective function could then be optimized, while ensuring that the schedule risk does not exceed the bound. Perhaps more importantly, they show that constraints containing an uncontrollable time-point can be converted to a set-bounded contingent link, thus drawing analogies between PSTNs and STNUs. Their solution uses Boole's inequality to bound above the risk and therefore is a conservative approximation of the true robustness. Wang et al. [221] introduce a more efficient method of CC-PSTN scheduling based on conflict detection. In this approach, the problem is decomposed into smaller problems: the first allocates risk evenly across uncertain durations resulting in an STNU, while the latter checks the STNU for SC and returns conflicts to be added to the first model in the form of constraints.

In some instances, the risk required to enforce SC can be deemed too high. Yu et al. [103] extend the chance-constrained framework to the *Relaxable* CC-PSTN by permitting the use of soft constraints which can be relaxed. The relaxable CC-PSTN is solved by Yu et al. as per Wang et al. [221] using a nonlinear solver, combined with a conflict detection mechanism based on identification of negative cycles in STNUs.

All aforementioned methods make use of non-linear optimization solvers to solve SC of PSTNs. Such approaches are often difficult to solve and offer no guarantee of global optimality. By using Boole's inequality and forming piecewise linear approximations of the cumulative density function (CDF), Santana et al. [101] were the first to present a fully linear encoding of PSTN SC. This enabled solutions to be found online, using commercial off the shelf linear programming solvers. Lund et al. [102] also leverage linear programming solvers in their Static Robust Execution Algorithm (SREA). They iteratively pose and solve an LP using non-bounding approximations of the probability mass in the tails of the probability distributions. By iteratively solving SREA within a loop, they present a dynamic execution algorithm which can react to unexpected outcomes at execution, thus paving the way to PSTN dynamic controllability.

To the best of the authors' knowledge, all previous approaches to PSTN SC assume independence [13, 100–103], and either use Boole's inequality to bound above the risk [13, 101, 103], or solve a generic non-linear optimization problem [100, 221]. Using Boole's inequality permits the use of LPs, however it is not guaranteed to return the optimal solution maximizing robustness (see Section 5.4); procedures used to solve generic non-linear optimization problems offer no guarantee on either optimality or computational efficiency.

In Section 5.5, we show that PSTN SC can be modelled as a convex optimization problem enabling globally optimal, robust schedules to be found. We suggest one approach in Section 5.6, that has been employed in convex optimization literature. Rather than using piecewise linear approximations of the CDF which is not convex (see Santana [101]), we form inner approximations of the negative log of the CDF which is convex. These inner approximations are analogous to piecewise linear approximations in one dimension, however generalise to polyhedral approximations at higher dimensions. This makes it possible to approximate multivariate random variables and consequently consider correlations in the optimization. Every approach outlined in this section assumes independence of probabilistic constraints, making our approach the first to consider correlations.

5.3.2 Correlations in Scheduling

While assuming independence when solving scheduling problems improves tractability, it can often lead to inaccurate or sub-optimal solutions. This is particularly the case when strong correlations exist. Correlated uncertainty exists in many practical applications of scheduling algorithms, in this section we aim to elucidate this with reference to some examples.

Industrial Production In Zhang et al. correlated uncertainty is studied in production scheduling of ethylene plants [25]. The authors highlight that causal relations between upstream and downstream equipment in the process, as well as uncertainty in the demand and supply of resources can lead to correlations between consumption rates of different furnaces, with coefficients as high as 0.5. In a similar vein, Lu et al. [222] study the single machine scheduling problem and show that correlations may arise in job processing times, driven by factors such as shortage of raw materials, availability of staff and machine breakdowns.

Energy Networks In renewable energy networks, energy providers face the challenge of balancing energy generation to cope with demand. Renewable energy generation from different turbines is inherently uncertain and depends strongly on correlation factors such as wind speed [223]. The problem of scheduling when to activate the generators is denoted the energy and reserve dispatch scheduling problem. Xu et al. [26] study the energy and reserve dispatch problem and note that supply of energy from different turbines is strongly correlated, with coefficients as high as 0.9.

Construction Ökmen and Öztaş [224] focus on analysing risk in construction schedules and highlight that weather may affect which tasks can be performed when, or how long it takes to perform particular tasks. Wang et al. [225] also note that labor and site conditions contribute to correlations in activity durations. Maronati and Petrovic [226] note that construction activity durations of the same type (i.e. welding or concrete pouring) are often highly correlated. Likewise, Eiris Pereira and Flood [227] note that correlation in activity durations greater than 0.8 can result in the construction crew spending 7% of their time idle, as well as a 12% extension in project duration.

Vehicle Routing Park et al. [216] showed that correlations can exist in travel times between roads, with coefficients as high as 0.75. They note that traffic congestion in upstream roads can correlate with increased travel times in the near future. Conversely, Nicholson [228], highlights that negative correlation can occur in road networks as a result of drivers increasing speed to make up for delays caused by congestion. Bakach et al. [215] study vehicle routing problems under correlated uncertainty and show that considering correlations can result in a 13.76% reduction in solution make span. For delivery companies, whose profit is driven by how many customers can be served in a set time, considering correlation could yield a significant improvement in profits.

5.4 Motivating Example

We motivate our approach by discussing the toy example given in Figure 5.4. Consider a drone used in the transportation of medical supplies. After being

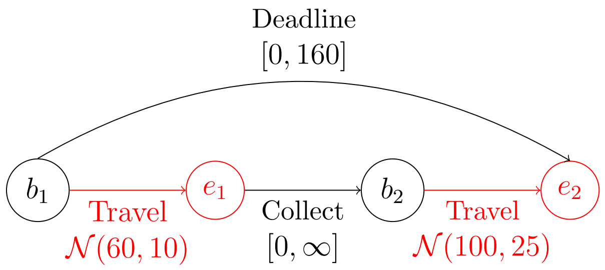


Figure 5.4: Image showing toy example

notified of a potential delivery, the drone must fly from a depot, to a location to pick it up. The travel time of this leg of the journey ($e_1 - b_1$) is described by the random variable $X_1 \sim \mathcal{N}(60, 10)$. After it has collected the supplies it must fly to the drop off point and deliver the supplies within the required time-frame (between 0 and 160 minutes after setting out $e_2 - b_1$). This leg of the journey ($e_2 - b_2$) can be described by the random variable $X_2 \sim \mathcal{N}(100, 25)$.

We want to find the schedule that maximizes robustness Γ. We have two uncontrollable constraints $c(b_2, e_1)$ and $c(e_2, b_1)$. From Equations (5.1) and (5.2) we have:

$$c(b_2, e_1) \equiv 0 \leq b_2 - b_1 - X_1 \leq \infty$$
$$c(e_2, b_1) \equiv 0 \leq b_2 + X_2 - b_1 \leq 160$$

The only decision variable is the difference between the time assigned to b_2 and b_1. W.l.o.g. we assume $b_1 = 0$.

Boole's Inequality Using Boole's inequality we can formulate the objective as:

$$\max_{b_2}\{P(b_2 - \infty \leq X_1 \leq b_2) + P(-b_2 \leq X_2 \leq -b_2 + 160)\}$$

If we consider first that $b_2 = 75$ then we have:

$$(F_{X_1}(75) - F_{X_1}(-\infty)) + (F_{X_2}(85) - F_{X_2}(-75)) = 1.21$$

Next we consider that $b_2 = 67$ so we have:

$$(F_{X_1}(67) - F_{X_1}(-\infty)) + (F_{X_2}(93) - F_{X_2}(-67)) = 1.14$$

Using Boole's, $b_2 = 75$ is clearly the better schedule.

Joint Outcome with Independence If we consider the joint outcome with independence then the objective is:

$$\max_{b_2}\{P(b_2 - \infty \leq X_1 \leq b_2)P(-b_2 \leq X_2 \leq -b_2 + 160)\}$$

For $b_2 = 75$:

$$(F_{X_1}(75) - F_{X_1}(-\infty))\,(F_{X_2}(85) - F_{X_2}(-75)) = 0.26$$

And $b_2 = 67$:

$$(F_{X_1}(67) - F_{X_1}(-\infty))\,(F_{X_2}(93) - F_{X_2}(-67)) = 0.30$$

Considering the joint outcome, the optimal solutions are switched with the schedule $b_2 = 67$, offering a 15% increase in robustness versus the solution returned using Boole's inequality. This effect is observed in greater detail in Figure 5.5.

Joint Outcome with Correlation We will now show that, even considering the joint outcome with independence is not necessarily guaranteed to return optimal solutions if the correlation is experienced when the schedule is executed.

We return to the drone example and consider that the travel times X_1 and X_2 are correlated due to uncertainty in wind speed. We plotted the robustness for varying b_2 and varying correlation coefficient ρ in Figure 5.5. Note that positive correlation could occur when the two legs of the drones journey are in

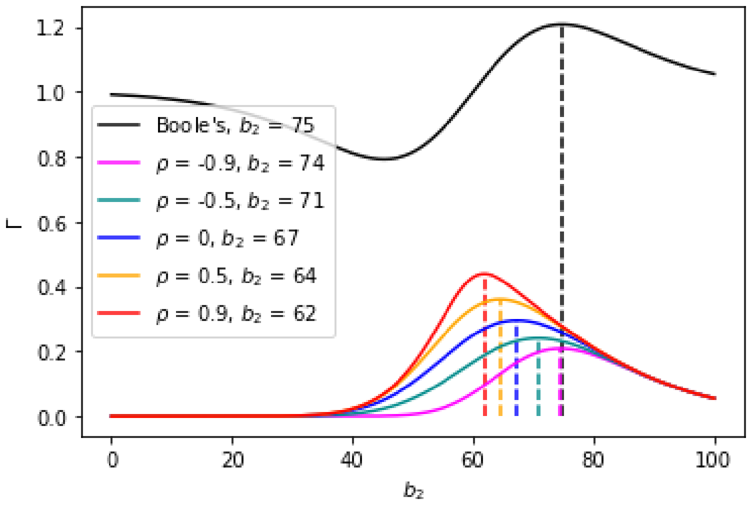


Figure 5.5: Comparison of robustness using Boole's versus actual robustness with varying correlation coefficient ρ (below).

the same direction (if one leg encounters a headwind, then the other leg should also encounter a headwind). On the other hand, negative correlation could occur if the two legs are in opposite directions (if one leg encounters a headwind, the other should encounter a tailwind and consequently take longer). Considering a fixed schedule of $b_2 = 67$, with $\rho = 0$ we have independence and consequently the robustness is as per the previous section $\Gamma = 0.30$. On the other hand, if the two variables have correlation $\rho = 0.9$ then the robustness $\Gamma = 0.39$. If we were to assume independence in the optimization then this is the best robustness we can hope for. It is clear from Figure 5.5, that better robustness can be achieved through scheduling b_2 five minutes earlier ($b_2 = 62$) such that the robustness $\Gamma = 0.44$. In this case, considering correlation in the scheduling process offers a 12.8% improvement in robustness.

To explain this difference, we refer to Figure 5.6, which shows a contour plot of the joint probability density function of $\boldsymbol{X} = [X_1, X_2]$. With $b_2 = 67$, the robustness is given by the volume beneath the contour plot within a rectangle with dimensions $-\infty \leq X_1 \leq 67$ and $-67 \leq X_2 \leq 93$ (shown by the vertical and horizontal blue lines). The dimensions of the box are fixed by the constant bounds, $[l_c, u_c]$ associated with each uncontrollable constraint. Finding the schedule that optimizes robustness, involves moving the box (by changing the schedule), such that it covers as much of the probability mass as possible. This in turn is dependent on the underlying distribution - for which correlation may have a significant effect. For the correlated case the optimal occurs at $b_2 = 62$, such that the rectangle has bounds $-\infty \leq X_1 \leq 62$ and $-62 \leq X_2 \leq 98$ (shown by the vertical and horizontal red lines).

Objective Accuracy It's worth noting that if you assume independence, you are not guaranteed to have a conservative estimate of the actual robustness. For example in Figure 5.5, with $\rho = 0$, the robustness from the model would be $\Gamma = 0.3$. The decision maker would be expected to make a decision based on this value, whereas in reality the robustness experienced could be much lower. If correlation $\rho = -0.9$ was experienced at execution time, the actual robustness from the schedule $b_2 = 67$ would be $\Gamma = 0.16$. Referring to Figure 5.6 can offer some insight into this effect. When we solve assuming independence, the solution is a conservative approximation of the probability mass under the blue contour plot, enclosed within the blue box. The actual robustness is the probability mass under the red contour plot (also within the blue box). This is not guaranteed to be strictly less than the equivalent probability mass under the independent probability density function. While the optimal schedule for Boole's and correlation $\rho = -0.9$ are similar, the objective of Boole's $\Gamma = 1.2$, offers nothing to a decision maker who has to reason over the risk of the schedule. Under such circumstances it becomes necessary to consider the correlation directly. We now formally introduce the Corr-STN:

Definition 11 (Corr-STN). *A Corr-STN is a tuple, $S^C = \langle T_c, T_u, C, D, R \rangle$, where T_c, T_u and C are as per the PSTN. R is the set of correlations involv-*

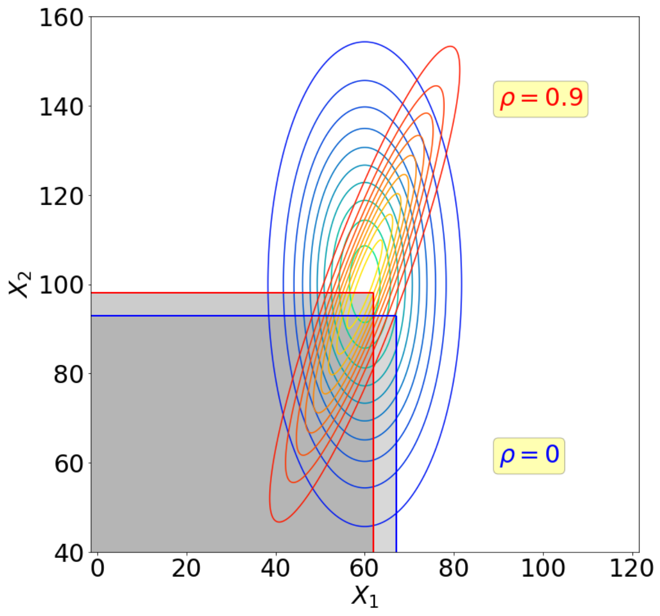


Figure 5.6: Image showing bi-variate Gaussian distribution probability density function with and without correlation.

ing a number of probabilistic constraints with correlation matrix ϱ. Each $r \in R$ defines an n dimensional multivariate Gaussian vector $\boldsymbol{X} \sim (\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. The set D is the set of independent probabilistic constraints. If there are no correlations, then the set $R = \varnothing$ and the set D is the set of all independent probabilistic constraints as per the PSTN.

5.5 Corr-STN SC is Convex

In this section we show how Corr-STN SC can be formulated as a convex optimization problem.

Decision Variables The decision variables include the vector of controllable time-points $\boldsymbol{x}$, and the vector of lower and upper bounds $\boldsymbol{z}$ (introduced below).

Controllable Constraints The controllable constraints can be written in the form of two less than inequalities: $b_j - b_i \leq u_{c,ij}$ and $b_i - b_j \leq -l_{c,ij}$, such that they represent a polyhedron: $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{\beta}$, where $\boldsymbol{A}$ is the coefficient matrix of values, $A_{ij} \in \{-1, 1, 0\}$, $\boldsymbol{\beta}$ is the vector of bounds $\beta_i \in \{u_c, -l_c\}$ and $\boldsymbol{x}$ is the decision variable vector.

Independent Probabilistic Constraints For each $d \in D$, we have a number of uncontrollable constraints preceding/succeeding it. From (5.3) to (5.6), we can write the uncontrollable constraints in the form: $b_i - b_j + l_{c,ij} \leq X_j \leq b_i - b_j + u_{c,ij}$ and $b_j - b_i - u_{c,ij} \leq X_i \leq b_j - b_i - l_{c,ij}$. Consequently, we have the matrix inequality $\boldsymbol{z}_d \leq \boldsymbol{T}_d\boldsymbol{x} + \boldsymbol{q}_d$, where $z_{d,i} \in \{u_d, -l_d\}$, $T_{d,ij} \in \{-1, 1, 0\}$ and $q_{d,i} \in \{u_c, -l_c\}$. The probability that the constraint is satisfied is given by the probability function $F_d = P(\boldsymbol{l}_d \leq \boldsymbol{X} \leq \boldsymbol{u}_d)$.

Correlations For each correlation $r \in R$, we write the n_r uncontrollable constraints involved in the correlation in the form: $\boldsymbol{z}_r \leq \boldsymbol{T}_r\boldsymbol{x} + \boldsymbol{q}_r$. The difference here is that we have more than one random variable involved in each correlation. The vector of upper and lower bounds are in the form $\boldsymbol{u}_r = [u_{r,1}, \ldots, u_{r,n_r}]^T$ and $\boldsymbol{l}_r = [l_{r,1}, \ldots, l_{r,n_r}]^T$, and therefore calculating the probability that the constraints are satisfied involves calculating the joint probability function $F_r = P(\boldsymbol{l}_r \leq \boldsymbol{X} \leq \boldsymbol{u}_r)$, for the multivariate distribution $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ as per Definition 11.

Objective Function For each $d \in D$ and $r \in R$, we know that the lower and upper bounds $\boldsymbol{l}_d, \boldsymbol{u}_d$ and $\boldsymbol{l}_r, \boldsymbol{u}_r$ are directly encoded in the vectors $\boldsymbol{z}_d$ and $\boldsymbol{z}_r$ which represent rows of a vector $\boldsymbol{z}$, such that $\boldsymbol{z} = [\boldsymbol{z}_{d_1}, \ldots, \boldsymbol{z}_{d_{|D|}}, \boldsymbol{z}_{r_1}, \ldots, \boldsymbol{z}_{r_{|R|}}]^T$. The objective function is to maximize the robustness Γ, giving the following optimization problem:

$$\max_{\boldsymbol{x},\boldsymbol{z}} \{ \prod_{r \in R} F_r \prod_{d \in D} F_d \mid \boldsymbol{z} \leq \boldsymbol{T}\boldsymbol{x} + \boldsymbol{q},\ \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{\beta} \}$$

On Convexity Following from above, the only non-linear function here is the robustness $\Gamma = \prod_{r \in R} F_r \prod_{d \in D} F_d$, used in the objective.

Proposition 1 (Corr-STN SC is Convex)**.** *Let $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ be the multi-variate Gaussian random vector representing the random durations. Prèkopa [33] shows that the multi-variate Gaussian distribution is log-concave. Note that we can rewrite $P(\boldsymbol{l} \leq \boldsymbol{X} \leq \boldsymbol{u})$, as $P(\boldsymbol{\eta X} \leq \boldsymbol{z})$, where $\boldsymbol{\eta} = [\boldsymbol{I}, -\boldsymbol{I}]^T$ and $\boldsymbol{z} = [\boldsymbol{u}, -\boldsymbol{l}]^T$ [229]. If $\boldsymbol{X}$ is log-concave, then so is $\boldsymbol{\xi} = \boldsymbol{\eta X}$, since it is a linear transformation of a log-concave distribution (Prèkopa [33] Theorem 10.2.4). If $\boldsymbol{\xi}$ is log-concave, then the function $P(\boldsymbol{\xi} \leq \boldsymbol{z})$ is log-concave since it is the cumulative probability function of a log-concave distribution (Prèkopa [33] Theorem 10.2.1). This implies that the functions F_r and F_d are log-concave. If F_r and F_d are log-concave then so is Γ since it is the product of log-concave functions [230] and therefore:*

$$\log\left(\prod_{r \in R} F_r \prod_{d \in D} F_d\right) = \sum_{r \in R} \log F_r + \sum_{d \in D} \log F_d$$

is concave. As a result we can reformulate the optimization problem as a convex one:

$$\min_{\boldsymbol{x},\boldsymbol{z}} \{\sum_{r \in R} \phi_r + \sum_{d \in D} \phi_d \mid \boldsymbol{z} \leq \boldsymbol{Tx} + \boldsymbol{q},\ \boldsymbol{Ax} \leq \boldsymbol{\beta}\} \tag{5.7}$$

In fact, the result in Proposition 1 is not unique to multi-variate Gaussian distributions: the result stands so long as the random vectors $\boldsymbol{X}$ have log-concave probability distributions. Many useful distributions contain this characteristic, a survey of which is provided by Bagnoli et al. [113].

Running Example Consider the small Corr-STN from Figure 5.4. In this example, we have no controllable constraints, no independent probabilistic constraints and only one correlation defining a multivariate Gaussian distribution $\boldsymbol{X} = [X_1, X_2]$. The uncontrollable constraints associated with the correlation are

$c(b_2, e_1)$ for X_1 and $c(e_2, b_1)$ for X_2 respectively. We encode this as:

$$\begin{bmatrix} u_{r,1} \\ u_{r,2} \\ -l_{r,1} \\ -l_{r,2} \end{bmatrix} \leq \begin{bmatrix} -1 & 1 \\ 1 & -1 \\ 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 160 \\ \infty \\ 0 \end{bmatrix}$$

The objective is to find the value of vectors:

$$\boldsymbol{x} = [b_1, b_2]^T$$
$$\boldsymbol{z} = [u_{r,1}, u_{r,2}, -l_{r,1}, -l_{r,2}]^T$$

that minimizes ϕ_r.

5.6 Method

The result of the previous section is that the problem is convex, allowing use of a rich suite of existing solution methods. For a recent survey of techniques, we refer to Van Ackooij [231]. In the coming section, we introduce one such method that forms an inner approximation of the convex functions ϕ_d and ϕ_r using a number of generated approximation points (hereby referred to as columns) [29, 232]. The problem is then solved via the column generation procedure as outlined in Algorithm 2, in which two optimization phases are iteratively solved:

1. The **Restricted Master Problem** (RMP), which solves the probability maximisation problem using the columns generated so far (line 4).
2. A **Column Generation Problem** (CGP) for each function ϕ_d and ϕ_r, which finds the best new column to include in the RMP (line 7).

The results of the RMP and CGP are stored in modelR and modelC respectively. We extract the dual values (modelR.duals) from the solution to the RMP, and use them to model the *reduced cost* which we set as the objective to the CGP (modelC.objective). Since we are minimizing reduced cost, any column whose reduced cost is negative is called an *improving* column. If an improving column can be found (line 8), we set terminate to False (line 9), and add the new column (line 10). The process then repeats until no improving column can be found.

Algorithm 2: Algorithm for SC of Corr-STN

```
   Input  : A Corr-STN, $S^C$
   Output: An optimization model $modelR$ containing a schedule that
           optimizes robustness.
1  $columns \leftarrow$ getInitialColumn($S^C$);
2  $terminate \leftarrow$ False;
3  while $terminate \leftarrow False$ do
4      $modelR \leftarrow$ RMP($columns, S^C$);
5      $terminate \leftarrow$ True;
6      for $function \in D \cup R$ do
7          $modelC \leftarrow$ CGP($modelR$.duals, $function$);
8          if $modelC$.objective $< 0$ then
9              $terminate \leftarrow$ False;
10             $columns$.add($modelC$.solution)
11         end
12     end
13 end
   Return: $modelR$
```

Inner Approximation Note that ϕ_r and ϕ_d are the only nonlinear functions in Equation (5.7) and they are convex. For any convex function ϕ, if we have enumerated sufficiently many finite points, $\{\boldsymbol{z}^1, \boldsymbol{z}^2, ..., \boldsymbol{z}^K\}$, referred to as *base* (see Geoffrion [233]), in its domain, and let $\phi^i := \phi(\boldsymbol{z}^i)$, then we can approximate $\min \phi(\boldsymbol{z})$ with :

$$\min\{\sum_{i=1}^{K} \phi^i \lambda^i : \sum_{i=1}^{K} \lambda^i = 1, \lambda^i \geq 0\} \tag{5.8}$$

where K depends on the desired level of approximation (see Figure 5.7). Note that from henceforth the notation ^{i} refers to the i^{th} column. Since the inner approximation is an over-estimate of the convex functions ϕ_r and ϕ_d, it is a conservative approximation of the robustness F_r and F_d.

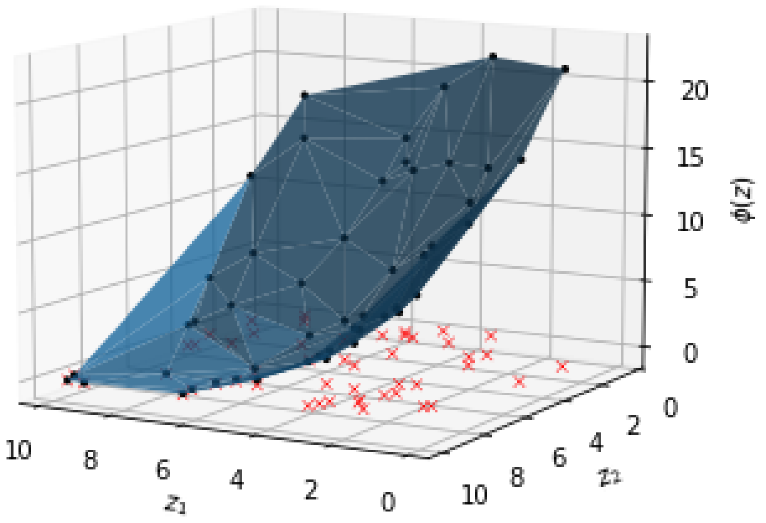


Figure 5.7: Inner approximation for a bivariate convex function $\phi(\boldsymbol{z})$. The red crosses are the approximation points (columns) $\boldsymbol{z}^i$ at which the function has been evaluated and the black dots are the function evaluations ϕ^i.

Running Example To highlight this we return to the running example. Assume that we have evaluated the function ϕ_r at a number of points: $\boldsymbol{l}^i, \boldsymbol{u}^i$ for $i = 1, 2, .., K$, such that ϕ_r^i refers to $\phi_r(\boldsymbol{l}^i, \boldsymbol{u}^i)$. The inner approximation would be:

$$\begin{bmatrix} u_{r,1}^1 & \cdots & u_{r,1}^K \\ u_{r,2}^1 & \cdots & u_{r,2}^K \\ -l_{r,1}^1 & \cdots & -l_{r,1}^K \\ -l_{r,2}^1 & \cdots & -l_{r,2}^K \end{bmatrix} \begin{bmatrix} \lambda_r^1 \\ \vdots \\ \lambda_r^K \end{bmatrix} \leq \begin{bmatrix} -1 & 1 \\ 1 & -1 \\ 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 160 \\ \infty \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 \ldots 1 \end{bmatrix} \begin{bmatrix} \lambda_r^1 \\ \vdots \\ \lambda_r^K \end{bmatrix} = 1, \quad \lambda_r^i \geq 0, \quad \phi_r = \sum_{i=1}^{K} \phi_r^i \lambda_r^i \tag{5.9}$$

While the column: $\boldsymbol{z}^i = [u_{r,1}^i, u_{r,2}^i, -l_{r,1}^i, -l_{r,2}^i]$.

$$\min_{\boldsymbol{x},\boldsymbol{\lambda}} \sum_{r\in R}\sum_{i=1}^{K_r} \phi_r^i \lambda_r^i + \sum_{d\in D}\sum_{i=1}^{K_d} \phi_d^i \lambda_d^i$$

$$\begin{aligned}
s.t.\ & \boldsymbol{Ax} \leq \boldsymbol{\beta} & \\
& \sum_{i=1}^{K_r} \lambda_r^i \boldsymbol{z}_r^i \leq \boldsymbol{T}_r\boldsymbol{x} + \boldsymbol{q}_r & r \in R\ (dual : \pi_r) \\
& \sum_{i=1}^{K_d} \lambda_d^i \boldsymbol{z}_d^i \leq \boldsymbol{T}_d\boldsymbol{x} + \boldsymbol{q}_d & d \in D\ (dual : \pi_d) \\
& \sum_{i=1}^{K_r} \lambda_r^i = 1 & r \in R\ (dual : \nu_r) \\
& \sum_{i=1}^{K_d} \lambda_d^i = 1 & d \in D\ (dual : \nu_d) \\
& x_i,\ \lambda^i \geq 1 &
\end{aligned}$$

Figure 5.8: Master Problem

Restricted Master Problem If we form an inner approximation for each independent probabilistic constraint and correlation using Equation (5.8), we can re-write Equation (5.7) in its approximate form as a linear program as shown in Figure 5.8. We refer to this as the **Master Problem** (MP). Notice that we have replaced the $\boldsymbol{z}$ variables in Equation (5.7) with $\boldsymbol{\lambda}$ variables. With each point, $\boldsymbol{z}^i$, in our base we can associate a *column* of coefficients in the constraint matrix corresponding to the variable λ^i (as shown in Equation (5.9)). The value of the $\boldsymbol{\lambda}$ variables allow for the convex combination of the columns enumerated so far. We refer K_r, K_d as the number of columns generated for each correlation r and independent probabilistic constraint d. Likewise we refer $\boldsymbol{\lambda}_r$ as the K_r dimensional vector of variables associated with the columns of a correlation r and $\boldsymbol{\lambda}_d$ as the K_d dimensional vector of variables associated with the columns generated for an independent probabilistic constraint d.

K can be prohibitively large when solving the MP in Figure 5.8 directly.

In addition, we are only interested in the columns from our base that are in the vicinity of the optimal solution. The key idea is that we iteratively solve a restricted version, that we refer to as the **Restricted Master Problem** (RMP), where only a subset, $\{\boldsymbol{z}^1, \ldots, \boldsymbol{z}^k\}$ such that $k << K$, of our base is considered. Note that the optimal solution to the RMP is always feasible to the MP. For it to be optimal, none of the unconsidered columns in our base would improve the objective when included in the RMP. If no such column exists, then the optimal solution of the RMP is also optimal to the MP.

Finding an Initial Feasible Point In order to initialise the algorithm, we must find an initial column $\boldsymbol{z}^0$, for which the RMP has a feasible solution. Any previous PSTN SC algorithm can be used to generate a feasible point. To see this, consider that we obtain a schedule, i.e. an assignment of a value to all the controllable time-points: $\boldsymbol{x}^0 \in \mathbb{R}^n_+$, which satisfies all the constraints. The equivalent column can then be evaluated as $\boldsymbol{z}^0 = \boldsymbol{T}\boldsymbol{x}^0 + \boldsymbol{q}$. Many efficient PSTN SC algorithms exist capable of finding such a feasible point, for details on how to implement such an algorithm, we refer the reader to the relevant paper [13, 100–102]. In this chapter, we chose to use the algorithm of Santana et al. [101] since it is an LP and can be solved efficiently.

Column Generation Problem A column is said to be an *improving* column if the reduced cost is negative [234]. Given a linear program $\min_{\boldsymbol{x}}\{\boldsymbol{c}^T\boldsymbol{x} \mid \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b},\ x_i \geq 0\}$, any variable x_i that takes a zero value in the simplex procedure is known as a *non-basic variable.* The *reduced cost* of introducing a non-basic variable x_i into the simplex basis is: $c_i - \boldsymbol{A}^{i^T}\boldsymbol{y}$, where $\boldsymbol{A}^i$ refers to column i of matrix A and $\boldsymbol{y}$ is the dual vector.

Running Example Returning to the ongoing example, we show how to generate an improving column $\boldsymbol{z}_r^{k+1}$, for ϕ_r. The objective coefficient would be

$c_{k+1} = \phi_r^{k+1}$. From Equation (5.9), the column of coefficients in the constraint matrix associated with variable λ^{k+1} is $\boldsymbol{A}^{k+1} = [\boldsymbol{z}_r^{k+1}, 1]^T$ and from Figure 5.8 the dual vector $y = [\boldsymbol{\pi}_r, \boldsymbol{\nu}_r]$. We can therefore find the best improving column by minimizing reduced cost, which amounts to solving the following optimization problem.

$$\begin{aligned} \min_{\boldsymbol{z}_r}\{\phi_r(z_r) - \boldsymbol{z}_r^T\boldsymbol{\pi}_r - \nu_r\} &:= \min_{\boldsymbol{l_r},\boldsymbol{u_r}}\{-\log F(\boldsymbol{l_r}, \boldsymbol{u}_r) \\ &-[\boldsymbol{u}_r, -\boldsymbol{l}_r]\begin{bmatrix}\boldsymbol{\pi}_{u_r}\\ \boldsymbol{\pi}_{l_r}\end{bmatrix} - \nu_r \mid \boldsymbol{u}_r > \boldsymbol{l}_r\} \end{aligned} \tag{5.10}$$

Such that $\boldsymbol{\pi}_{u_{r,i}} = \sum_{\{j:z_{r,j}=u_{r,i}\}} \pi_{r,j}$ refers to element i in vector $\boldsymbol{\pi}_{u_r}$, where $j = 1, 2, ..., m$ and m is the number of rows in constraint matrix $\boldsymbol{z_r} \leq \boldsymbol{T}_r\boldsymbol{x} + \boldsymbol{q}_r$. Similarly $\boldsymbol{\pi}_{l_{r,i}} = \sum_{\{j:z_{r,j}=-l_{r,i}\}} \pi_{r,j}$ refers to element i in vector $\boldsymbol{\pi}_{l_r}$. To prevent domain errors with $\log(0)$, we constrain the upper bound to be greater than the lower bound, $\boldsymbol{u_r} > \boldsymbol{l}_r$.

We refer to this as the **Column Generation Problem** (CGP). We solve one CGP for all $r \in R$ and $d \in D$. If no improving column can be found for any function, at any iteration, then it is not possible to find another variable λ_{k+1} (and column $\boldsymbol{z}^{k+1}$) which will improve the objective when entered into the simplex basis. In other words, our inner approximation already contains the optimal solution and so we can terminate the algorithm. The convergence of this procedure has been shown in Dantzig [234].

In order to solve Equation (5.10), it is necessary to efficiently compute the gradient vector. This can be evaluated as:

$$\nabla\left(\phi(\boldsymbol{z}_r) - \boldsymbol{z}_r^T\boldsymbol{\pi}_r - \nu_r\right) = -\frac{\nabla F(\boldsymbol{l}_r, \boldsymbol{u}_r)}{F(\boldsymbol{l}_r, \boldsymbol{u}_r)} - \begin{bmatrix}\boldsymbol{\pi}_{u_r}\\ \boldsymbol{\pi}_{l_r}\end{bmatrix}$$

There exists efficient algorithms capable of calculating cumulative probabilities of multivariate Gaussian distributions (e.g. [235]). Prekopa [33] proves it is possible to analytically evaluate the gradient of the function $F(z)$ for multi-variate Gaussian distributions using the same efficient algorithm [236, 237]. Van Ackooij et al. [236] present a formula for the case, $\nabla F(\boldsymbol{l}, \boldsymbol{u})$ which relies on the same result.

On Correlated Chance Constrained STNs In this chapter we focus on problems in which the objective it to optimize robustness. Often it is the case that some other objective function $\boldsymbol{c}^T\boldsymbol{x}$, should be optimized subject to a user-defined tolerance on risk [13]. As an example we may want to deliver all medicines while minimizing the number of trips used (since this may be a proxy for cost), subject to the constraint that the plan has a 95% chance of succeeding. In this section we show that this approach can easily be adapted to solve a chance-constrained problem while considering correlation.

We define an allowable tolerance on risk δ such that we can model the chance constraint as: $1 - \prod_{r\in R} F_r \prod_{d\in D} F_d \leq \delta$. Taking the log of both sides as per Section 5.5 and substituting $\phi_\delta = \log(\delta - 1)$ we have:

$$\sum_{r\in R} \phi_r + \sum_{d\in D} \phi_d \leq \phi_\delta$$

Again, we are free to form an inner approximation of the functions ϕ_r and ϕ_d as outlined in Equation (5.8). The only difference versus the probability maximization case, is that the reduced cost function needs some minor modification. The objective is now some arbitrary function $c^T x$ and so the objective coefficient associated with each new approximation point is zero. Furthermore we now have a dual variable associated with the chance constraint which we will call μ. The column generation problem then becomes:

$$\min_{\boldsymbol{z}_r}\{-\mu\phi_r(\boldsymbol{z}_r) - \boldsymbol{z}_r^T\boldsymbol{\pi}_r - \nu_r\}$$

which relies on exactly the same functions and derivatives and so can be solved using the same approach.

5.7 Experimental Setup

We experimentally validated our approach on a number of Corr-STNs generated for the drone delivery planning domain introduced in Section 5.4. In this section we discuss the generation of the Corr-STN instances.

5.7.1 Planning Domain and Problem

To generate Corr-STN instances, a temporal planning domain and problems were first constructed using PDDL [238].

Description of PDDL Domain

The domain file is provided in Appendix A and contains information describing the rules of how the world works.

In the drone planning domain, we have a set of drones which must deliver a set of medicines to their respective delivery points. Here, we have two classes of objects, those which can be located at a place, denoted *locatables* and the locations themselves. The locatables contain the drones as well as the medicines to be delivered.

The drone can perform a number of actions to achieve the goal of delivering the medicine. First it can *move* between two locations. For this action to take place, the drone must be located at the initial location, the two locations must be connected and the drone must have sufficient battery to complete the journey. The battery used for the journey is a function of the battery rate and the travel time between the two locations.

When the battery of the drone reduces below a certain level, it may wish to *recharge* the batteries. Only some of the locations contain charging facilities, which we refer to as the depots. Once the recharge action has taken place, the battery of the drone is refilled to its full capacity. The time it takes to complete this action depends on the recharge rate of the drone and the battery level remaining.

If the drone is at the same location as a medicine and the medicine is not too heavy, then it may wish to *pick-up* the medicine. Each medicine has a fixed weight, and each drone has a load capacity which defines the maximium weight it can carry. Conversely, if the drone is carrying a medicine then it can *drop-off* the medicine at a given location, providing that it is located at that location.

Finally, to model the expiration deadline of the medicines, we create an additional action *complete-delivery*, which is conditional on: the medicine being at the location and the medicine not being expired. Timed intitial literals (TIL), were used to model the expiration of each medicine. TILs enforce that a predicate becomes true at a given time. In this instance, the medicines become expired when their expiration time is reached. The effect of the complete-delivery action is that the medicine is considered *delivered* at the location.

PDDL Problem Generation

Each PDDL problem involved a number of deliveries which must be completed by the drones. To generate problems of different size, we varied the number of drones and medicines to be delivered. Ten cases were generated for each combination of drones from the set $\{1, 2, 3, 4\}$, and medicines from the set $\{1, 2, 4, 8\}$.

Each drone was sampled from three sizes: small, medium and large, with varying load capacity, battery capacity, battery rate and recharge rate, as outlined in Table 5.1. Similarly, each medicine delivery was sampled from 9 different medicine types with varying weight, probability and expiration time as per Table 5.2.

Table 5.1: Drone types

Size	**Load Capacity**	**Battery Capacity**	**Battery Rate**	**Recharge Rate**
Small	10	50	1	10
Medium	20	100	1	5
Large	50	150	1	4

Table 5.2: Medicine Types

Name	**Weight**	**Expiration Time**	**Probability**
Penicillin	2	400	0.25
Insulin	1	180	0.15
Defibrillator	20	100	0.05
Blood	10	120	0.15
Organ	20	100	0.1
Vaccine	2	150	0.1
Atorvastatin	2	200	0.05
Levothyroxine	3	300	0.05
Metformin	5	500	0.1

The number of locations was kept constant at 10, with the number of depots containing charging facilities fixed at 2. The depots were uniformly sampled for each problem instance from the 10 locations available. Finally, the distances between each location was sampled in the range of 10 to 100.

Each drone and medicine was then randomly assigned to one of the 10 locations. Thus the initial state was composed of the drones being located at their given locations, drones having full battery and load capacity, medicines being located at their given locations and not currently expired.

The goal was to complete the delivery of all medicines to their respective delivery location. Each delivery location was also randomly sampled from the available locations. One example problem file is provided in Appendix B for reference.

5.7.2 Corr-STN Instance Generation

Each PDDL problem instance was then solved using the temporal planner OPTIC [239], resulting in a temporal plan. The plan output for the PDDL problem file in Appendix B is shown in Listing 5.1.

Listing 5.1: Example of PDDL plan file.

```
0.000: (move d0 l7 l4) [20.000]
20.001: (move d0 l4 l8) [20.000]
40.002: (move d0 l8 l5) [30.000]
70.002: (pick-up d0 l5 m0) [5.000]
75.002: (move d0 l5 l8) [30.000]
105.003: (move d0 l8 l3) [30.000]
135.004: (drop-off d0 l3 m0) [5.000]
140.005: (complete-delivery m0 l3) [0.001]
```

An STN was then constructed from each PDDL plan file using the Open Task Planning Library[1], as shown in Figure 5.9.

For each STN, we then generated 10 separate PSTN instances by sampling mean and standard deviations to apply to actions. For each PSTN, we then create 3 separate Corr-STN instances by sampling random correlation matrices of size 2, 3 and 4. Finally, TIL deadlines were then varied to generate Corr-STN problems with a wide variety of robustness. The result was a total of 5850 Corr-STN problem instances of which 4872 could be solved (the unsolvable ones had a robustness of zero)[2].

[1] `https://github.com/taskplanning/otpl`

[2] Source code and benchmark problems can be accessed online at: `https://anonymous.4open.science/r/CORRSTN-3E78/`

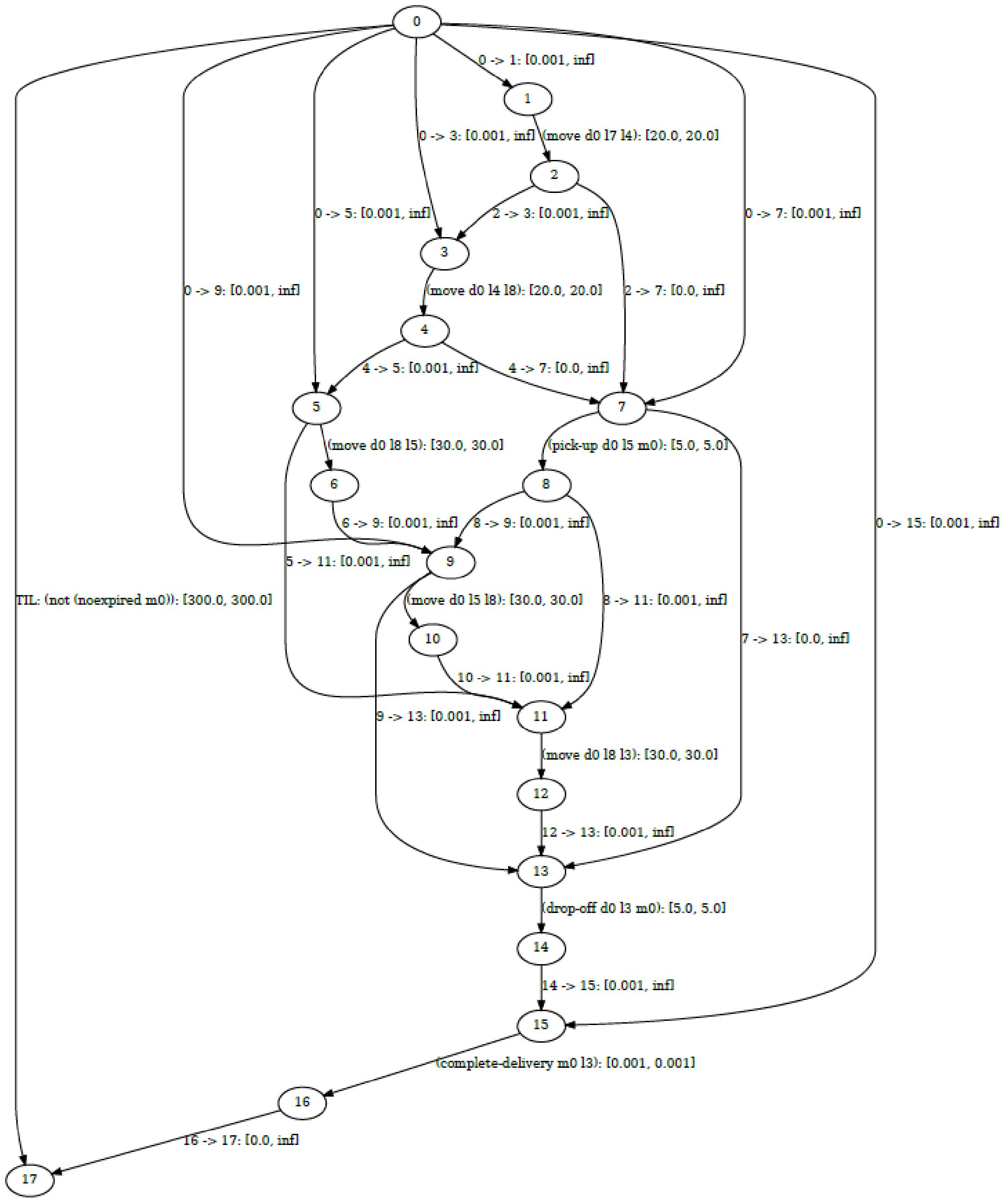


Figure 5.9: Image showing STN example constructed from plan.

5.7.3 Solution Methods

Each Corr-STN was then solved using three methods: an SC Linear Program with Boole's inequality (implementation of the PARIS algorithm from Santana et al. [101]) (referred to as **Boole's**); Column Generation method assuming independence (referred to as **Independent**); Column Generation with correlation (referred to as **Correlated**). Python was used for the implementation with Gurobi as the linear programming optimizer, and the SLSQP solver within the Python package SciPy as the column generation solver.

5.8 Results

Plots showing results for robustness, runtime and accuracy are provided in Figures 5.10, 5.11 and 5.12 respectively. Note that Γ^{MC} and Γ^{TH} refer to the Monte Carlo robustness, and theoretical robustness obtained from the optimization model. The notation b, i, c, $c2$, $c3$ and $c4$ refer to the results for Boole's, independent, all correlated cases, and correlated cases of sizes 2, 3 and 4 respectively (number of events that are considered correlated with one another). Figures 5.10 and 5.12 are plotted against the optimal probability, as obtained from the Monte Carlo simulations considering correlation.

Robustness To assess the robustness we simulated each schedule, for each Corr-STN, 20,000 times and calculated the Monte-Carlo robustness. Boole's and independent robustness were then compared to correlated, and the percentage difference plotted in Figure 5.10.

The improvement in robustness when we consider correlation versus using Boole's is substantial but has a wide variance. In general, we see a significant improvement on problems in which the optimal robustness is low with some cases offering up to 80% improvement. The intuition for this can be gained from

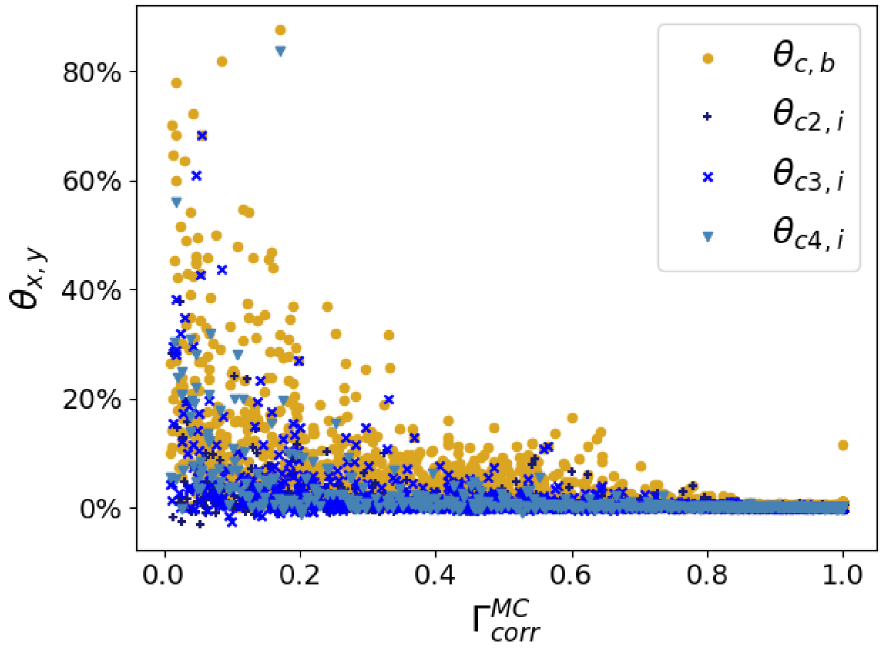


Figure 5.10: Plot showing % difference in Monte Carlo robustness for different solutions: $\theta_{x,y} = (\Gamma^{MC}_x - \Gamma^{MC}_y)/\Gamma^{MC}_x \times 100$. Boole's and independent are compared to correlated of different sizes.

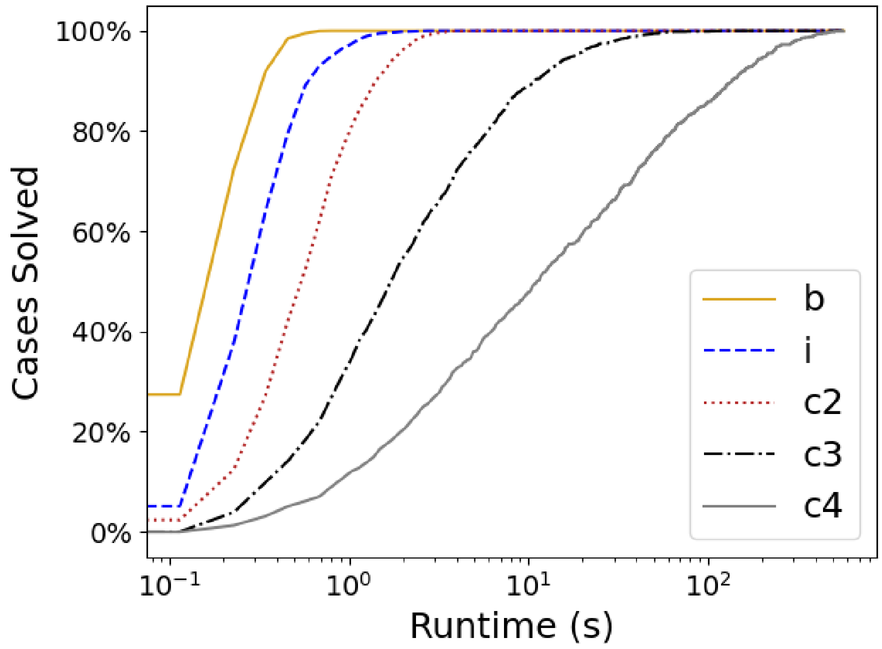


Figure 5.11: Plot showing % of cases solved versus runtime for Boole's, independent and correlated.

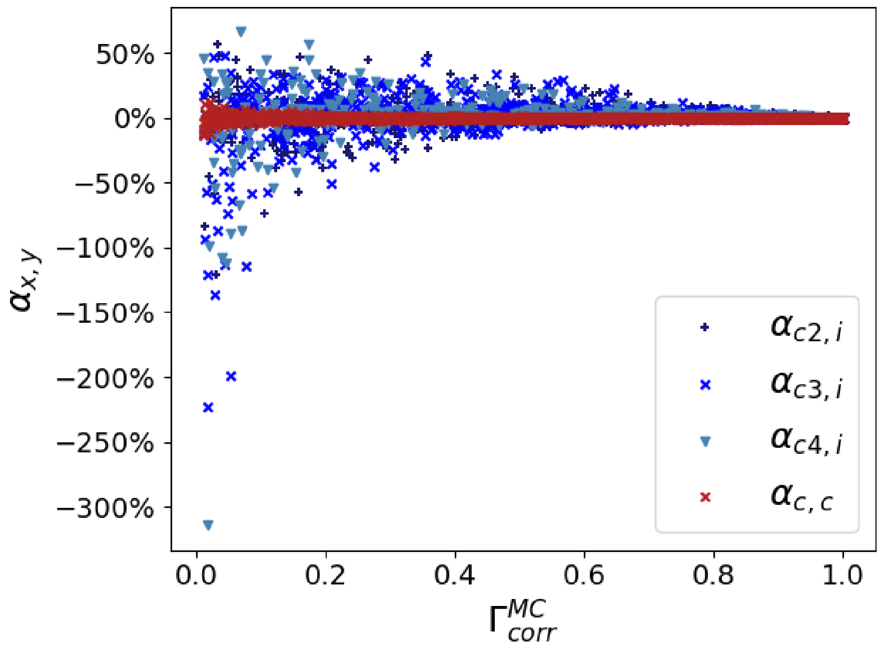


Figure 5.12: Plot showing % difference in Monte Carlo and theoretical robustness: $\alpha_{x,y} = (\Gamma^{MC}_x - \Gamma^{TH}_y)/\Gamma^{MC}_x \times 100$. Theoretical for independent and correlated are compared to correlated Monte-Carlo.

Figure 5.6. As discussed in Section 5.4, the problem of Corr-STN SC involves moving a box (by varying the schedule) of n dimensions over the n dimensional multivariate Gaussian probability density function. When the box is small and constrained to the outer corners of the distribution, the optimal location of the box can be quite different. Of the 928 cases where the correlated Monte Carlo robustness was less than 0.5, correlated offered a mean improvement of 8.51% over Boole's and 3.50% over independent.

Runtime Figure 5.11 plots the percentage of cases solved versus runtime for the Boole's, independent and correlated of varying correlation sizes. As expected using the Boole's LP is substantially faster with all cases solved within 1 second. This is a result of the fact that the encoding is entirely linear. On the other hand approximately 90% of the independent cases and 80% of the correlated size 2 cases could be solved within 1 second. The runtime grows exponentially with the size of the distribution, some cases with correlation size 4 took up to 400 seconds. Nonetheless approximately 80% of the correlated size 3 cases and 50% of the correlated size 4 cases could be solved within 10 seconds.

Note that the stopping criteria allows for a trade-off between the solution quality and runtime. All of the problems were solved with a gap of 1% (i.e. $\varepsilon = 0.01$), however it is possible to terminate the algorithm earlier, and return the best solution found so far. Generally, we found that the column generation procedure tends to struggle to close the gap for some instances, hence those which took significantly longer than other cases of a comparable size, tended to reach a reasonable intermediate solution quite quickly. Of course, the amount of time that it is acceptable to dedicate to the scheduling depends on the application domain, for example a robot domain may require very fast solutions, whereas a crew scheduling domain may not. It's also important to mention, that since we initialise the column generation procedure with the solution using Boole's inequality, the column generation solution is at least as good as the Boole's inequality solution from the start of the procedure.

Accuracy To measure accuracy, we compare the theoretical robustness obtained from the objective for independent and correlated, with the Monte-Carlo robustness considering the correlation. The percentage difference is plotted in Figure 5.12. If we assume independence, we can obtain theoretical robustness values which are up to 3 times higher than the actual robustness observed through Monte-Carlo simulation. This is because assuming independence is not guaranteed to provide a bounding approximation of the correlated robustness (see Section 5.4 and Figure 5.6). We do not plot the theoretical versus experimental robustness using Boole's since the objective is not representative of the actual probability.

Comparison on other Domains In order to show the applicability of our approach to other domains, we provide a comparison of robustness Γ, and runtime t, for Boole's, independent and correlated on International Planning Competition [240] domains: rovers and crew-planning. We solved 10 instances from each domain and generated 3 Corr-STNs for each instance. To highlight the percentage improvement in robustness for low versus high robustness cases, we generated 3 instances from each STN associated with low, mid and high robustness (identified in the Robustness Level column). Correlation size 2 was selected as it offered a good improvement in robustness for minimal impact in runtime. Similarly, we add correlation coefficients of 0.9 across constraints involved in the correlations, as high correlation instances were shown to result in the greatest improvement in robustness of correlated versus Boole's and independent. Standard deviations were randomly sampled to apply to actions as per the drone domain. Results for the rovers domain is provided in Table 5.3, whereas results for the crew-planning domain is provided in Table 5.4. Note that $\theta_{c,b}$ and $\theta_{c,i}$ refers to the percentage difference in robustness of correlated versus Boole's and independent (as outlined at the start of this section) and that #Cts refers to the number of constraints involved in the network.

First, it is important to note that the trend observed in the drone planning domain (greater percentage improvement in robustness when the overall robustness is low) is also observed in the rovers and crew-planning domain. In general, considering correlation appears to be more effective on the rovers domain versus the crew-planning domain. This is thought to be attributed to the difference in structure of the two domains. Whereas rovers problem instances tend to involve consecutive actions to achieve the goals (as per the drone planning problem), the crew-planning domain often involves more concurrent actions occurring. Nonetheless, we still see a significant improvement in robustness in both domains. In the rovers domain we observe an average improvement of 20.61%, 7.56% and

Instance	Robustness Level	# Cts	Γ_b^{MC}	Γ_i^{MC}	Γ_c^{MC}	t_b (s)	t_i (s)	t_c (s)	$\theta_{c,b}$ (%)	$\theta_{c,i}$ (%)
1	Low	44	0.041	0.042	0.061	0.177	0.289	1.523	32.869	30.574
1	Mid	44	0.377	0.445	0.467	0.177	0.296	1.494	19.244	4.594
1	High	44	0.948	0.949	0.950	0.182	0.346	2.411	0.237	0.158
2	Low	26	0.106	0.120	0.155	0.134	0.246	0.677	31.266	22.739
2	Mid	26	0.515	0.526	0.547	0.132	0.217	0.662	5.982	3.891
2	High	26	0.952	0.953	0.954	0.131	0.221	1.476	0.283	0.100
3	Low	64	0.378	0.404	0.412	0.332	0.514	1.403	8.172	1.858
3	Mid	64	0.892	0.895	0.897	0.335	0.502	1.157	0.547	0.167
3	High	64	0.957	0.959	0.960	0.336	0.456	4.584	0.323	0.099
4	Low	28	0.169	0.161	0.170	0.158	0.268	1.153	0.881	5.316
4	Mid	28	0.510	0.566	0.580	0.153	0.258	1.605	12.131	2.473
4	High	28	0.905	0.907	0.907	0.156	0.254	1.237	0.210	0.033
5	Low	94	0.001	0.002	0.002	0.405	0.699	2.302	29.730	16.216
5	Mid	94	0.249	0.260	0.270	0.397	0.734	2.709	7.765	3.558
5	High	94	0.788	0.790	0.794	0.403	0.733	3.107	0.768	0.497
7	Low	73	0.213	0.208	0.232	0.379	0.440	1.702	8.523	10.375
7	Mid	73	0.588	0.589	0.605	0.372	0.446	1.098	2.908	2.644
7	High	73	0.873	0.882	0.886	0.374	0.554	2.955	1.372	0.350
8	Low	143	0.009	0.014	0.016	0.658	1.174	4.098	42.188	14.688
8	Mid	143	0.366	0.373	0.381	0.646	1.100	4.627	4.051	2.281
8	High	143	0.930	0.935	0.936	0.649	0.976	12.753	0.662	0.112
10	Low	152	0.018	0.020	0.023	0.706	1.127	2.644	21.245	14.592
10	Mid	152	0.195	0.206	0.223	0.706	1.075	3.615	12.371	7.297
10	High	152	0.907	0.913	0.916	0.706	1.144	4.317	0.939	0.289
11	Low	145	0.110	0.119	0.136	0.644	1.036	4.274	18.950	12.266
11	Mid	145	0.593	0.604	0.613	0.641	0.890	3.310	3.302	1.459
11	High	145	0.891	0.901	0.902	0.638	0.973	2.643	1.175	0.122
12	Low	91	0.010	0.010	0.012	0.461	0.709	2.619	12.340	13.617
12	Mid	91	0.353	0.370	0.381	0.468	0.679	1.785	7.284	2.809
12	High	91	0.949	0.952	0.953	0.474	0.705	8.324	0.430	0.058

Table 5.3: Table showing results for rover domain.

Instance	Robustness Level	# Cts	Γ_b^{MC}	Γ_i^{MC}	Γ_c^{MC}	t_b (s)	t_i (s)	t_c (s)	$\theta_{c,b}$ (%)	$\theta_{c,i}$ (%)
1	Low	86	0.219	0.219	0.235	0.068	0.112	0.275	6.729	6.729
1	Mid	86	0.739	0.746	0.755	0.069	0.134	1.237	2.191	1.271
1	High	86	0.995	0.995	0.995	0.068	0.322	0.116	0.000	0.000
2	Low	73	0.216	0.215	0.232	0.090	0.129	0.821	7.272	7.293
2	Mid	73	0.405	0.405	0.423	0.091	0.134	0.620	4.291	4.291
2	High	73	0.841	0.849	0.852	0.092	0.435	1.210	1.274	0.340
3	Low	82	0.110	0.110	0.121	0.111	0.168	1.192	8.838	8.838
3	Mid	82	0.284	0.284	0.300	0.112	0.167	0.623	5.343	5.343
3	High	82	0.849	0.858	0.860	0.111	0.536	0.651	1.296	0.302
4	Low	116	nan	0.052	0.056	nan	0.362	1.503	100.000	8.363
4	Mid	116	0.620	0.620	0.625	0.136	0.244	0.533	0.832	0.832
4	High	116	0.929	0.929	0.931	0.142	0.200	1.512	0.204	0.204
5	Low	89	0.052	0.052	0.058	0.155	0.697	0.643	9.801	9.801
5	Mid	89	0.656	0.664	0.673	0.154	0.264	1.091	2.556	1.330
5	High	89	0.898	0.904	0.904	0.157	0.303	1.218	0.658	0.044
6	Low	133	0.041	0.041	0.043	0.178	0.870	1.601	5.665	5.665
6	Mid	133	0.585	0.593	0.601	0.183	0.308	1.260	2.638	1.248
6	High	133	0.896	0.900	0.900	0.187	0.342	2.119	0.455	0.056
7	Low	160	0.045	0.045	0.049	0.141	0.211	0.597	7.536	7.434
7	Mid	160	0.783	0.784	0.790	0.140	0.278	0.923	0.855	0.665
7	High	160	0.958	0.958	0.958	0.135	0.214	0.943	0.026	0.031
8	Low	142	0.039	0.039	0.043	0.157	0.238	0.757	9.618	9.618
8	Mid	142	0.249	0.249	0.255	0.159	0.238	0.929	2.312	2.312
8	High	142	0.950	0.951	0.951	0.156	0.280	1.400	0.131	0.074
9	Low	158	0.012	0.012	0.013	0.222	0.326	0.996	11.364	11.364
9	Mid	158	0.466	0.467	0.479	0.222	0.414	1.593	2.661	2.556
9	High	158	0.718	0.723	0.730	0.231	0.450	3.339	1.644	0.891
11	Low	212	0.092	0.098	0.108	0.245	0.501	1.640	14.444	8.843
11	Mid	212	0.213	0.223	0.230	0.245	0.479	2.782	7.599	3.300
11	High	212	0.996	1.000	1.000	0.250	0.271	3.684	0.360	0.005

Table 5.4: Table showing results for crew-planning domain.

0.64% over Boole's and 14.22%, 3.12% and 0.18% improvement over independent for low, mid and high robustness respectively. In the crew planning domain we observe an average improvement of 18.13%, 3.13% and 0.60% over Boole's and 8.39%, 2.31% and 0.19% over independent for low, mid and high robustness respectively. All rover instances considering correlation are solved within 12.753s, whereas all crew-planning problems are solved within 3.684s.

5.9 Conclusion

To summarise, we formally define the Corr-STN and show that Corr-STN SC is convex for multivariate (log-concave) distributions. We introduce one solution method using column generation, in which we iteratively refine and optimize on an approximation of the distributions.

In our experimental validation we solved a number of Corr-STNs using 1. Boole's inequality, 2. column generation with independence and 3. column generation with correlations of varying sizes. We compared schedules in terms of robustness, runtime and accuracy. We find that, for problems in which the optimal probability is small, considering correlation can offer a significant robustness improvement versus Boole's inequality and column generation with independence. Since these cases have a high probability of failure, it may be worth spending additional time to ensure that the schedule has the best chance of succeeding. We generalise this result by comparing our approach versus Boole's and independent on a number of International Planning Competition domains. To ensure an accurate, bounding approximation of robustness, considering correlation is necessary, however it comes with additional computational expense which may be prohibitive for many applications. Nonetheless, the any-time nature of our approach means that the decision maker can spend as long as they want finding a schedule: the more time spent, the better the solution will be.

While we have empirically shown that considering the correlation in the scheduling process can be important and have outlined a method for doing so, we note that the value of considering the correlation varies significantly. For some cases, the improvement is substantial, however for others it is not worth the additional computational effort. There are a vast number of problem specific factors which affect this: size/magnitude of correlation, tightness of constraints as well as which constraints we consider correlated to name a few. In future work we hope to define a metric which can be used to determine the benefit of considering the correlation for particular networks.

Chapter 6

Conclusion

Planning and scheduling are necessary functions for autonomous agents to be able to act rationally to achieve their goals. Due to the increased use of AI in sensitive applications, developing planning and scheduling models which are robust to uncertainty is becoming increasingly important. In this thesis we tackled robustness from a probabilistic perspective, and developed approaches to solving planning and scheduling problems with robustness guarantees using the column generation method.

Our conclusions are presented in two sections. First, we summarise the key insights and contributions presented in this thesis. After this, we briefly mention a number of potential directions for future work related to the thesis.

6.1 Summary and Contributions

In Chapter 2, we reviewed, and then classified past approaches for achieving robustness in planning and scheduling according to proactive, reactive and probabilistic. Taking a proactive approach and considering ways to account for the uncertainty in advance can prevent delays at execution time. In reality however, it is not possible to consider in advance all possible outcomes that may arise in the real world. Reacting to issues dynamically as they arise may be preferable in some cases, however it can lead to delays at execution time. Furthermore, sensitive applications may require a numeric guarantee on *how* robust the solution is. This can only be achieved by explicitly modelling the uncertainty using probabilities, and then solving an optimization problem to reason over the uncertainty. By reviewing techniques for handling uncertainty in optimization, we motivate the use of the column generation method.

In Chapter 3, we introduced the reader to the column generation method. We mention that a number of other excellent sources are available outlining the fundamentals of column generation (perhaps my favourite being Desrosiers and Lübbecke's "a primer in column generation" [241]), however our overview differs in that it is intended to give the reader an *intuition* for how it works. This is achieved through the use of numerous figures and examples. The justification for this, was to make it easier for readers to apply the technique to solving new problems, as has been the core focus of this thesis.

In Chapter 4, we introduced a column generation approach to solving the VNF-PRP that amounts to iteratively solving the RMP, which places VNFs onto the network and routes the SFCs; and a CGP for each SFC that generates a new improving path. We highlight that prior approaches to this problem typically only handle a subset of the features required for 5G network slicing. By treating QoS metrics such as latency, data-rate or availability as the objective, they do not distinguish between the different levels required for each slice use case. Most past approaches use either exact ILPs which are not scalable; or meta-heuristics which give no guarantee of solution quality. While column generation can find bounded-optimal solutions to practical sized problems, past attempts at using it to solve the VNF-PRP contain numerous issues: they assume that VNF instances can be fractionally split in terms of CPU and RAM, they do not consider replication, assume that the routing flow is known a priori and do not model availability. In this chapter our contribution is as follows:

- The first VNF placement model capable of computing a full placement, routing and replication solution while providing guarantees on solution quality.
- The first VNF-PRP model capable of handling throughput, latency and availability QoS constraints, while maximizing QoS. This makes it the first approach capable of satisfying the diverse service requirements expected in 5G and beyond.
- We show that the VNF-PRP for network slicing can be solved via a column generation procedure in which the sub problem is a shortest path problem on an augmented network, enabling the application of efficient solution methods.
- We experimentally show that our VNF-PRP model is capable of finding near optimal solutions on a realistic MEC network test case within a reasonable time-frame.

In Chapter 5, we formally introduced the definition of Corr-STN and presented a column generation approach to Corr-STN SC. This amounts to iteratively solving the RMP, which finds the optimal schedule given an approximation of the joint distribution; and a CGP, which finds a new point to refine the approximation. Past approaches to solving the problem of PSTN SC assume independence and either use Boole's inequality to bound above the risk, or solve a generic non-linear optimization problem. Assuming independence is not guaranteed to find the most robust schedule, or even return a conservative approximation of the robustness if correlations are experienced at execution time. Using Boole's can be overly conservative and grossly inaccurate, whereas solving a non-linear optimization problem can be computationally inefficient and is not guaranteed to find the global optimum. In this chapter our contribution is as follows:

- We introduce and define for the first time the Corr-STN and the problem of Corr-STN SC. This makes it possible to model many practical scheduling problems containing correlations.
- We show that Corr-STN SC can be framed as a convex optimization problem, when the uncertain durations are modelled using a multivariate Gaussian distribution.
- We present an inner approximation approach to solving Corr-STN SC using column generation. This is the first algorithm for solving Corr-STN SC.
- We empirically show that prior approaches to PSTN SC assuming independence are not guaranteed to give a conservative approximation of robustness. Likewise, approaches leveraging Boole's inequality can be grossly inaccurate.
- Our experimental validation shows that by considering correlations, our Corr-STN SC algorithm finds strictly more robust solutions versus prior PSTN SC algorithms.

6.2 Future Research Directions

Planning and Scheduling for the Unknown Unknowns In Chapter 2, we reviewed methods for achieving robustness in planning and scheduling and mentioned the importance of robustness guarantees. It's worth mentioning that in order to derive a robustness guarantee, we need to be able to accurately capture the uncertainty using probabilities. Where sufficient data is available, this can be leveraged to derive or predict a distribution. Going forward it will be important to develop models which are robust to all uncertainty, not just that which we have sufficient data to model or predict. This is highlighted by Dietterich [242] who commented on the use of AI models in high stakes applications:

"Such applications require AI methods to be robust to both the known unknowns (those uncertain aspects of the world about which the computer can reason explicitly) and the unknown unknowns (those aspects of the world that are not captured by the system's models)"

The author goes on to discuss a number of approaches to dealing with the latter. Model failure prediction can be used to detect when a model has insufficient knowledge before it makes a mistake. Chang and Frank [243] use an online plan viability checker for autonomous robot task planning in space. The plan viability checker is able to continuously monitor changes to the world and determine conditions that lead to a partially executed plan no longer being valid, thus initiating a re-planning procedure. Using an ensemble of AI models [244] can help to improve robustness, since it introduces diversity to the decision making. Fern and Lewis [245] demonstrate this in a planning context by using an ensemble of Monte Carlo tree search models to make planning decisions in a variety of domains. Ensemble reinforcement learning models have also shown promise in planning [246, 247].

Improving the Efficiency of Decomposition Methods In Chapter 5 we present an approach to stochastic optimization in which the sub problem involves computing multi-variate probabilities and gradients. This procedure is particularly time consuming. Recently there has been a great deal of focus on improving the speed of decomposition methods using heuristics and surrogate models.

Ruszczyński and Świtanowski [248] present numerous techniques for accelerating Benders decomposition to solve stochastic network design problems. By leveraging a combination of cutting planes, partial decomposition, heuristics, stronger cuts, reduction and warm-start strategies they are able to achieve significant run time improvements. Lee et al. [249] proposed using a machine learning regressor and classifier to predict practically useful cuts. Rather than using machine learning in place of the sub problem, Mana et al. [250] use a reinforcement learning surrogate model in place of the master problem, resulting in a 30% reduction in run time. In column generation, Yu et al. [251] use a hybrid column generation algorithm, in which the CGP is solved using a number of meta heuristics. Finally Kraul et al. [252] use machine learning to predict the optimal dual variables for instances of the cutting stock problem. In general, the use of machine learning technologies to improve the efficiency of decomposition methods such as the column generation method has shown promising results, however it's worth noting that this is only the case if a similar problem will be solved repeatedly, else the training time is prohibitive.

Dynamic Virtual Network Function Placement and Routing In Chapter 4, our approach produced a static solution to the VNF-PRP. In reality, a number of events can occur which result in a violation of the SLA [253, 254]. A dynamic VNF-PRP solution requires the capability of predicting SLA violations and computing a plan to reconfigure the network so as to avoid the violation. It's also worth mentioning that the arrival time of SLA violations is uncertain but can often be predicted using historical data. It may be the case that the new configuration is more costly than the latter, hence it is beneficial to delay the reconfiguration as long as possible. On the other hand, if the SLA violation is encountered, the ISP may be responsible for paying a premium.

Hence, a dynamic VNF-PRP may involve: planning a sequence of actions to reconfigure the network so as to satisfy the SLAs and scheduling the actions so as to minimize cost subject to some tolerance on risk. We envision such a problem could be solved using a combination of the techniques from Chapters 4 and 5. The column generation procedure in Chapter 4 could be warm started using the pre-existing paths, alongside some additional constraints to generate a new optimal state. This goal state could then be passed to a planner to compute the sequence of actions required to achieve it; and a CC-PSTN SC problem could be formulated and solved using the plan, to decide when to schedule the actions while trading off against operational expenditure and SLA violation cost.

Correlated Dynamic Controllability In Chapter 5, we tackled the problem of Corr-STN SC, but did not attempt to solve the problem of Corr-STN dynamic controllability. Dynamic controllability enables the agent to leverage information gained at execution time to improve robustness. Dynamic controllability of PSTNs is an ongoing research problem but has been addressed recently in a number of papers [102, 255, 256].

From a Corr-STN perspective, each time an uncertain outcome is observed we have access to new information which can be used to inform our strategy. For example, consider two sequential actions whose durations are positively correlated. If we observe that the first action takes longer than expected, then the correlation tells us we can expect that the latter uncertain duration will also take longer. Thus our schedule can be adjusted taking into account this new information.

Combined Temporal and Resource Controllability It's worth mentioning that correlated uncertainty can also exist in resources. Considering correlations across resources and durations could be achieved through the use of a Correlated Simple Temporal Network with Resources (Corr-STNR).

We mention that temporal networks with resources are not new - Laborie introduced them in 2003 [257] and Combi et al. [258] address resource availability in the context of conditional STNUs. Cashmore et al. [259] compute the set of all parameters for which a temporal plan is valid - including resource parameters such as consumption rates. Extending Corr-STNs to handle resources could be achieved by deriving a joint distribution over the set of all uncertain parameters, as opposed to just action durations.

Appendix A

PDDL Drone Delivery Domain

```
(define (domain drone-delivery)
    (:requirements :typing :durative-actions :fluents :timed-initial-literals)
    (:types
        location locatable - object
        drone medicine - locatable
    )
    (:predicates
        (noexpired ?m - medicine)
        (delivered ?m - medicine ?l - location)
        (located-at ?o - locatable ?l - location)
        (connected ?l1 ?l2 - location)
        (carrying ?d - drone ?m - medicine)
        (is-depot ?l - location)
        (nocharging ?d - drone)
        (noloading ?d - drone)
    )
    (:functions
        (load-capacity ?d - drone)
        (weight ?m - medicine)
        (battery-capacity ?d - drone)
        (battery-level ?d - drone)
        (battery-rate ?d - drone)
        (recharge-rate ?d - drone)
        (travel-time ?l1 ?l2 - location)
    )
```

```
(:durative-action move
     :parameters ( ?d - drone ?l1 ?l2 - location)
     :duration (= ?duration (travel-time ?l1 ?l2))
     :condition (and
          (at start(located-at ?d ?l1)) (over all(connected ?l1 ?l2))
          (over all (nocharging ?d)) (over all (noloading ?d))
          (at start(>= (battery-level ?d)
              (* (battery-rate ?d) (travel-time ?l1 ?l2)))))
     :effect (and
          (at start(not (located-at ?d ?l1))) (at end (located-at ?d ?l2))
          (at end (decrease (battery-level ?d)
              (* (battery-rate ?d) (travel-time ?l1 ?l2)))))
  )
  (:durative-action recharge
     :parameters (?d - drone ?l - location)
     :duration (= ?duration (/ (- (battery-capacity ?d)
                 (battery-level ?d)) (recharge-rate ?d)))
     :condition (and
          (over all (located-at ?d ?l)) (over all (is-depot ?l))
          (over all (noloading ?d)))
     :effect(and
          (at start (not (nocharging ?d))) (at end (nocharging ?d))
          (at end (assign (battery-level ?d) (battery-capacity ?d))))
  )
  (:durative-action pick-up
     :parameters (?d - drone ?l - location ?m - medicine)
     :duration (= ?duration 5)
     :condition (and
          (over all (located-at ?d ?l)) (at start (located-at ?m ?l))
          (at start (> (load-capacity ?d) (weight ?m)))
          (over all (nocharging ?d)))
     :effect (and
          (at start (not (noloading ?d)))
          (at start (decrease (load-capacity ?d) (weight ?m)))
          (at start (not (located-at ?m ?l))) (at end (carrying ?d ?m))
          (at end (noloading ?d)))
  )
```

```
    (:durative-action drop-off
        :parameters (?d - drone ?l - location ?m - medicine)
        :duration (= ?duration 5)
        :condition (and
            (at start (located-at ?d ?l)) (at start (carrying ?d ?m))
            (over all (nocharging ?d)))
        :effect (and
            (at start (not (noloading ?d))) (at start (not (carrying ?d ?m)))
            (at end (located-at ?m ?l)) (at end (noloading ?d))
            (at start (increase (load-capacity ?d) (weight ?m))))
    )
    (:action complete-delivery
        :parameters (?m - medicine ?l - location)
        :precondition (and
            (noexpired ?m) (located-at ?m ?l))
        :effect (and
            (not (located-at ?m ?l)) (delivered ?m ?l))
    )
)
```

Appendix B

PDDL Drone Delivery Problem

```
(define (problem instance-1)
(:domain drone-delivery)
(:objects
    d0 - drone
    l0 l1 l2 l3 l4 l5 l6 l7 l8 l9 - location
    m0 - medicine
 )
(:init
;; depots
(is-depot l8) (is-depot l7)

;; drones
(located-at d0 l7) (noloading d0) (nocharging d0)
(= (load-capacity d0) 50) (= (battery-capacity d0) 150)
(= (battery-level d0) 150) (= (battery-rate d0) 1) (= (recharge-rate d0) 4)

;; medicines
(located-at m0 l5) (noexpired m0) (at 300 (not (noexpired m0))) (= (weight m0) 3)

;; locations
(connected l0 l7) (= (travel-time l0 l7) 70) (connected l0 l9) (= (travel-time l0 l9) 50)
(connected l1 l2) (= (travel-time l1 l2) 30) (connected l1 l6) (= (travel-time l1 l6) 30)
(connected l1 l7) (= (travel-time l1 l7) 20) (connected l1 l9) (= (travel-time l1 l9) 60)
(connected l2 l1) (= (travel-time l2 l1) 30) (connected l2 l6) (= (travel-time l2 l6) 60)
(connected l3 l4) (= (travel-time l3 l4) 80) (connected l3 l5) (= (travel-time l3 l5) 70)
(connected l3 l7) (= (travel-time l3 l7) 40) (connected l3 l8) (= (travel-time l3 l8) 30)
(connected l3 l9) (= (travel-time l3 l9) 50) (connected l4 l3) (= (travel-time l4 l3) 80)
(connected l4 l7) (= (travel-time l4 l7) 20) (connected l4 l8) (= (travel-time l4 l8) 20)
(connected l5 l3) (= (travel-time l5 l3) 70) (connected l5 l6) (= (travel-time l5 l6) 40)
(connected l5 l8) (= (travel-time l5 l8) 30) (connected l5 l9) (= (travel-time l5 l9) 30)
(connected l6 l1) (= (travel-time l6 l1) 30) (connected l6 l2) (= (travel-time l6 l2) 60)
(connected l6 l5) (= (travel-time l6 l5) 40) (connected l6 l7) (= (travel-time l6 l7) 30)
(connected l6 l9) (= (travel-time l6 l9) 80) (connected l7 l0) (= (travel-time l7 l0) 70)
(connected l7 l1) (= (travel-time l7 l1) 20) (connected l7 l3) (= (travel-time l7 l3) 40)
(connected l7 l4) (= (travel-time l7 l4) 20) (connected l7 l6) (= (travel-time l7 l6) 30)
(connected l8 l3) (= (travel-time l8 l3) 30) (connected l8 l4) (= (travel-time l8 l4) 20)
(connected l8 l5) (= (travel-time l8 l5) 30) (connected l9 l0) (= (travel-time l9 l0) 50)
(connected l9 l1) (= (travel-time l9 l1) 60) (connected l9 l3) (= (travel-time l9 l3) 50)
(connected l9 l5) (= (travel-time l9 l5) 30) (connected l9 l6) (= (travel-time l9 l6) 80)
)
(:goal
    (and (delivered m0 l3))
))
```

Bibliography

[1] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, “Network slicing in 5G: Survey and challenges,” *IEEE Communications Magazine*, vol. 55, no. 5, pp. 94–100, 2017.

[2] Q. Zhang, F. Liu, and C. Zeng, “Adaptive interference-aware VNF placement for service-customized 5G network slices,” in *IEEE International Conference on Computer Communications*, 2019, pp. 2449–2457.

[3] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, “Mastering the game of Go without human knowledge,” *Nature*, vol. 550, pp. 354–359, 2017.

[4] J. Tromp and G. Farnebäck, “Combinatorics of Go,” in *International Conference on Computers and Games.* Springer, 2006, pp. 84–99.

[5] D. Dietrich, C. Papagianni, P. Papadimitriou, and J. S. Baras, “Network function placement on virtualized cellular cores,” in *International Conference on Communication Systems and Networks*, 2017, pp. 259–266.

[6] A. Pozanco, K. Papasotiriou, D. Borrajo, and M. Veloso, “Combining heuristic search and linear programming to compute realistic financial plans,” in *International Conference on Automated Planning and Scheduling*, vol. 33, no. 1, 2023, pp. 527–531.

[7] S. Khan, S. Parkinson, M. Roopak, R. Armitage, and A. Barlow, "Automated planning to prioritise digital forensics investigation cases containing indecent images of children," in *International Conference on Automated Planning and Scheduling*, vol. 33, no. 1, 2023, pp. 500–508.

[8] M. Veale and F. Zuiderveen Borgesius, "Demystifying the draft EU artificial intelligence act—analysing the good, the bad, and the unclear elements of the proposed approach," *Computer Law Review International*, no. 4, pp. 97–112, 2021.

[9] N. A. Smuha, "The EU approach to ethics guidelines for trustworthy artificial intelligence," *Computer Law Review International*, vol. 20, no. 4, pp. 97–106, 2019.

[10] IEEE, "IEEE standard glossary of software engineering terminology," Standard 610.12, 1990.

[11] S. Chaudhuri, S. Gulwani, and R. Lublinerman, "Continuity and robustness of programs," *Communications of the ACM*, vol. 55, no. 8, pp. 107–115, 2012.

[12] M. Fox, R. Howey, and D. Long, "Exploration of the robustness of plans," in *AAAI Conference on Artificial Intelligence*, 2006, pp. 834–839.

[13] C. Fang, P. Yu, and B. C. Williams, "Chance-constrained probabilistic simple temporal problems," in *AAAI Conference on Artificial Intelligence*, 2014, pp. 2264–2270.

[14] D. Bertsimas, D. B. Brown, and C. Caramanis, "Theory and applications of robust optimization," *SIAM review*, vol. 53, no. 3, pp. 464–501, 2011.

[15] G. B. Dantzig and P. Wolfe, "Decomposition principle for linear programs," *Operations Research*, vol. 8, no. 1, pp. 101–111, 1960.

[16] P. C. Gilmore and R. E. Gomory, "A linear programming approach to the cutting-stock problem," *Operations Research*, vol. 9, no. 6, pp. 849–859, 1961.

[17] ——, "A linear programming approach to the cutting stock problem—part II," *Operations Research*, vol. 11, no. 6, pp. 863–888, 1963.

[18] J. Desrosiers, F. Soumis, and M. Desrochers, "Routing with time windows by column generation," *Networks*, vol. 14, no. 4, pp. 545–565, 1984.

[19] E. L. Lawler and D. E. Wood, "Branch-and-bound methods: A survey," *Operations Research*, vol. 14, no. 4, pp. 699–719, 1966.

[20] D. Soldani, F. Fadini, H. Rasanen, J. Duran, T. Niemela, D. Chandramouli, T. Hoglund, K. Doppler, T. Himanen, J. Laiho *et al.*, "5G mobile systems for healthcare," in *2017 IEEE 85th vehicular technology conference (VTC Spring)*. IEEE, 2017, pp. 1–5.

[21] M. Gidlund, T. Lennvall, and J. Åkerberg, "Will 5G become yet another wireless technology for industrial automation?" in *2017 IEEE International Conference on Industrial Technology (ICIT)*. IEEE, 2017, pp. 1319–1324.

[22] S. Hakak, T. R. Gadekallu, P. K. R. Maddikunta, S. P. Ramu, M. Parimala, C. De Alwis, and M. Liyanage, "Autonomous vehicles in 5G and beyond: A survey," *Vehicular Communications*, vol. 39, p. 100551, 2023.

[23] 3GPP, "Service requirements for the 5G system, rel. 15," Tech. Rep. 22.261, 2016.

[24] A. Chabrier, "Vehicle routing problem with elementary shortest path based column generation," *Computers & Operations Research*, vol. 33, no. 10, pp. 2972–2990, 2006.

[25] Y. Zhang, X. Jin, Y. Feng, and G. Rong, "Data-driven robust optimization under correlated uncertainty: a case study of production scheduling in ethylene plant," *Computers & Chemical Engineering*, vol. 109, pp. 48–67, 2018.

[26] X. Xu, Z. Yan, M. Shahidehpour, Z. Li, M. Yan, and X. Kong, "Data-driven risk-averse two-stage optimal stochastic scheduling of energy and reserve with correlated wind power," *IEEE Transactions on Sustainable Energy*, vol. 11, no. 1, pp. 436–447, 2019.

[27] A. Prékopa, "Probabilistic programming," *Handbooks in operations research and management science*, vol. 10, pp. 267–351, 2003.

[28] D. Dentcheva, B. Lai, and A. Ruszczyński, "Dual methods for probabilistic optimization problems," *Mathematical methods of operations research*, vol. 60, no. 2, pp. 331–346, 2004.

[29] C. I. Fábián, E. Csizmás, R. Drenyovszki, W. van Ackooij, T. Vajnai, L. Kovács, and T. Szántai, "Probability maximization by inner approximation," *Acta Polytechnica Hungarica*, vol. 15, no. 1, pp. 105–125, 2018.

[30] M. Ghallab, D. Nau, and P. Traverso, *Automated Planning: Theory and Practice.* Elsevier, 2004.

[31] M. L. Pinedo, *Scheduling.* Springer, 2012, vol. 29.

[32] S. Sinha, *Mathematical Programming: Theory and Methods.* Elsevier, 2005.

[33] A. Prékopa, *Stochastic Programming.* Springer Science & Business Media, 2013, vol. 324.

[34] J. Vermaelen, H. T. Dinh, and T. Holvoet, "A survey on probabilistic planning and temporal scheduling with safety guarantees," in *ICAPS Workshop on Planning and Robotics*, 2020.

[35] S. Bensalem, K. Havelund, and A. Orlandini, "Verification and validation meet planning and scheduling," *International Journal on Software Tools for Technology Transfer*, vol. 16, pp. 1–12, 2014.

[36] P. M. Verderame, J. A. Elia, J. Li, and C. A. Floudas, "Planning and scheduling under uncertainty: a review across multiple sectors," *Industrial & Engineering Chemistry Research*, vol. 49, no. 9, pp. 3993–4017, 2010.

[37] A. J. Davenport and J. C. Beck, "A survey of techniques for scheduling with uncertainty," *Unpublished manuscript. Available from http://tidel.mie.utoronto.ca/publications.php*, 2000.

[38] T. Chaari, S. Chaabane, N. Aissani, and D. Trentesaux, "Scheduling under uncertainty: Survey and research directions," in *IEEE International Conference on Advanced Logistics and Transport*, 2014, pp. 229–234.

[39] C. C. Marinagi, T. Panayiotopoulos, and C. D. Spyropoulos, "AI planning and intelligent agents," in *Intelligent Techniques for Planning.* IGI Global, 2005, pp. 225–258.

[40] C. Aeronautiques, A. Howe, C. Knoblock, I. D. McDermott, A. Ram, M. Veloso, D. Weld, D. W. SRI, A. Barrett, D. Christianson *et al.*, "PDDL the planning domain definition language," *Technical Report*, 1998.

[41] M. Fox and D. Long, "The third international planning competition: Temporal and metric planning." in *Artificial Intelligence Planning and Scheduling*, 2002, pp. 333–335.

[42] R. Dechter, I. Meiri, and J. Pearl, "Temporal constraint networks," *Artificial Intelligence*, vol. 49, no. 1-3, pp. 61–95, 1991.

[43] R. W. Floyd, "Algorithm 97: Shortest path," *Communications of the ACM*, vol. 5, p. 345, 1962.

[44] G. E. Vieira, J. W. Herrmann, and E. Lin, "Rescheduling manufacturing systems: a framework of strategies, policies, and methods," *Journal of Scheduling*, vol. 6, pp. 39–62, 2003.

[45] R. Howey and D. Long, "VAL's progress: The automatic validation tool for PDDL2. 1 used in the international planning competition," in *ICAPS Workshop on the International Planning Competition*, 2003, pp. 28–37.

[46] B. J. Clement and M. D. Johnston, "Design of a deep space network scheduling application," in *International Workshop on Planning and Scheduling for Space*, 2006, pp. 22–25.

[47] M. D. R-Moreno, G. Brat, N. Muscettola, and D. Rijsman, "Validation of a multi-agent architecture for planning and execution," in *International Workshop on Principles of Diagnosis*, 2007, pp. 368–371.

[48] J. Penix, C. Pecheur, and K. Havelund, "Using model checking to validate AI planner domain models," in *Annual Software Engineering Workshop, NASA Goddard*, 1998.

[49] F. Raimondi, C. Pecheur, and G. Brat, "PDVer, a tool to verify PDDL planning domains," in *ICAPS Workshop on Verification and Validatio of Planning and Scheduling Systems*, 2009.

[50] A. Goldberg, K. Havelund, and C. McGann, "Runtime verification for autonomous spacecraft software," in *IEEE Aerospace Conference*, 2005, pp. 507–516.

[51] R. P. Goldman and M. S. Boddy, "Conditional linear planning." in *Artificial Intelligence Planning and Scheduling*, 1994, pp. 80–85.

[52] D. S. Weld, C. R. Anderson, and D. E. Smith, "Extending graphplan to handle uncertainty & sensing actions," in *AAAI Conference on Artificial Intelligence*, 1998, pp. 897–904.

[53] R. P. Petrick and F. Bacchus, "A knowledge-based approach to planning with incomplete information and sensing." in *Artificial Intelligence Planning and Scheduling*, vol. 2, 2002, pp. 212–222.

[54] B. Bonet and H. Geffner, "Planning with incomplete information as heuristic search in belief space," in *International Conference on Artificial Intelligence Planning Systems*, 2000, pp. 52–61.

[55] P. Bertoli, A. Cimatti *et al.*, "Improving heuristics for planning as search in belief space." in *Artificial Intelligence Planning and Scheduling*, 2002, pp. 143–152.

[56] P. Bertoli, A. Cimatti, and M. Roveri, "Conditional planning under partial observability as heuristic-symbolic search in belief space," in *European Conference on Planning*, 2001, pp. 379–384.

[57] D. Bryce, S. Kambhampati, and D. E. Smith, "Planning graph heuristics for belief space search," *Journal of Artificial Intelligence Research*, vol. 26, pp. 35–99, 2006.

[58] J. Hoffmann and R. Brafman, "Contingent planning via heuristic forward search with implicit belief states," in *International Conference on Automated Planning and Scheduling*, 2005, pp. 71–80.

[59] A. Albore, H. Palacios, and H. Geffner, "A translation-based approach to contingent planning." in *International Joint Conference on Artificial Intelligence*, vol. 9, 2009, pp. 1623–1628.

[60] R. Brafman and G. Shani, “A multi-path compilation approach to contingent planning,” in *AAAI Conference on Artificial Intelligence*, vol. 26, no. 1, 2012, pp. 1868–1874.

[61] B. Bonet and H. Geffner, “Belief tracking for planning with sensing: Width, complexity and approximations,” *Journal of Artificial Intelligence Research*, vol. 50, pp. 923–970, 2014.

[62] C. Muise, V. Belle, and S. McIlraith, “Computing contingent plans via fully observable non-deterministic planning,” in *AAAI Conference on Artificial Intelligence*, vol. 28, no. 1, 2014.

[63] Y. Carreno, Y. Petillot, and R. P. Petrick, “Compiling contingent planning into temporal planning for robust AUV deployments,” in *ICAPS Workshop on Planning and Robotics*, 2021.

[64] M. Drummond, J. Bresina, and K. Swanson, “Just-in-case scheduling,” in *AAAI Conference on Artificial Intelligence*, vol. 94, 1994, pp. 1098–1104.

[65] S. V. Mehta, “Predictable scheduling of a single machine subject to breakdowns,” *International Journal of Computer Integrated Manufacturing*, vol. 12, no. 1, pp. 15–38, 1999.

[66] R. O’Donovan, R. Uzsoy, and K. N. McKay, “Predictable scheduling of a single machine with breakdowns and sensitive jobs,” *International Journal of Production Research*, vol. 37, no. 18, pp. 4217–4233, 1999.

[67] A. Davenport, C. Gefflot, and C. Beck, “Slack-based techniques for robust schedules,” in *European Conference on Planning*, 2014, pp. 7–18.

[68] A. Cimatti, M. Do, A. Micheli, M. Roveri, and D. E. Smith, “Strong temporal planning with uncontrollable durations,” *Artificial Intelligence*, vol. 256, pp. 1–34, 2018.

[69] R.-K. Li, Y.-T. Shyu, and S. Adiga, “A heuristic rescheduling algorithm for computer-based production scheduling systems,” *International Journal of Production Research*, vol. 31, no. 8, pp. 1815–1826, 1993.

[70] M. Yamamoto and S. Nof, “Scheduling/rescheduling in the manufacturing operating system environment,” *International Journal of Production Research*, vol. 23, no. 4, pp. 705–722, 1985.

[71] R. J. Abumaizar and J. A. Svestka, “Rescheduling job shops under random disruptions,” *International Journal of Production Research*, vol. 35, no. 7, pp. 2065–2082, 1997.

[72] J. Kuster, D. Jannach, and G. Friedrich, “Applying local rescheduling in response to schedule disruptions,” *Annals of Operations Research*, vol. 180, pp. 265–282, 2010.

[73] C. Bierwirth and D. C. Mattfeld, “Production scheduling and rescheduling with genetic algorithms,” *Evolutionary Computation*, vol. 7, no. 1, pp. 1–17, 1999.

[74] M. Fox, A. Gerevini, D. Long, and I. Serina, “Plan stability: Replanning versus plan repair,” in *International Conference on Automated Planning and Scheduling*, 2006, pp. 212–221.

[75] R. Van Der Krogt and M. De Weerdt, “Plan repair as an extension of planning.” in *International Conference on Automated Planning and Scheduling*, 2005, pp. 161–170.

[76] S. W. Yoon, A. Fern, and R. Givan, “FF-Replan: A baseline for probabilistic planning.” in *International Conference on Automated Planning and Scheduling*, 2007, pp. 352–359.

[77] S. S. Panwalkar and W. Iskander, “A survey of scheduling rules,” *Operations Research*, vol. 25, no. 1, pp. 45–61, 1977.

[78] P. Morris, “Dynamic controllability and dispatchability relationships,” in *International Conference on the Integration of Contstraint Programming, Artificial Intelligence and Operations Research.* Springer, 2014, pp. 464–479.

[79] M. Cairo and R. Rizzi, “Dynamic controllability made simple,” in *International Symposium on Temporal Representation and Reasoning*, 2017, p. 8:1–8:16.

[80] A. Cimatti, L. Hunsberger, A. Micheli, R. Posenato, and M. Roveri, “Dynamic controllability via timed game automata,” *Acta Informatica*, vol. 53, pp. 681–722, 2016.

[81] M. B. Do and S. Kambhampati, “Improving temporal flexibility of position constrained metric temporal plans.” in *International Conference on Automated Planning and Scheduling*, 2003, pp. 42–51.

[82] O. Lima, M. Cashmore, D. Magazzeni, A. Micheli, and R. Ventura, “Robust execution of deterministic plans in non-deterministic environments,” in *ICAPS Workshop on Integrated Planning, Acting and Execution*, 2020.

[83] A. Orlandini, A. Finzi, A. Cesta, and S. Fratini, “TGA-based controllers for flexible plan execution,” in *Annual German Conference on AI*, 2011, pp. 233–245.

[84] M. Fox, D. Long, and D. Magazzeni, “Plan-based policy-learning for autonomous feature tracking,” in *International Conference on Automated Planning and Scheduling*, vol. 22, 2012, pp. 38–46.

[85] N. Kushmerick, S. Hanks, and D. S. Weld, "An algorithm for probabilistic planning," *Artificial Intelligence*, vol. 76, no. 1-2, pp. 239–286, 1995.

[86] R. Bellman, "A Markovian decision process," *Journal of Mathematics and Mechanics*, vol. 6, pp. 679–684, 1957.

[87] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* John Wiley & Sons, 2014.

[88] B. Lacerda, F. Faruq, D. Parker, and N. Hawes, "Probabilistic planning with formal performance guarantees for mobile service robots," *International Journal of Robotics Research*, vol. 38, no. 9, pp. 1098–1123, 2019.

[89] G. E. Monahan, "State of the art—a survey of partially observable Markov decision processes: theory, models, and algorithms," *Management Science*, vol. 28, no. 1, pp. 1–16, 1982.

[90] A. Nilim and L. El Ghaoui, "Robust control of Markov decision processes with uncertain transition matrices," *Operations Research*, vol. 53, no. 5, pp. 780–798, 2005.

[91] E. Altman, *Constrained Markov Decision Processes.* Routledge, 2021.

[92] S. Thiébaux, B. Williams *et al.*, "RAO*: An algorithm for chance-constrained POMDP's," in *AAAI Conference on Artificial Intelligence*, 2016, pp. 3308–3314.

[93] F. Teichteil-Königsbuch, "Path-constrained Markov decision processes: bridging the gap between probabilistic model-checking and decision-theoretic planning." in *European conference on artificial intelligence*, 2012, pp. 744–749.

[94] H. L. Younes and M. L. Littman, "PPDDL1. 0: An extension to pddl for expressing planning domains with probabilistic effects," *Technical Report CMU-CS-04-162*, 2004.

[95] S. Sanner *et al.*, "Relational dynamic influence diagram language (RDDL): Language description," *Technical Report NICTA*, 2010.

[96] M. Steinmetz, J. Hoffmann, and O. Buffet, "Goal probability analysis in probabilistic planning: Exploring and enhancing the state of the art," *Journal of Artificial Intelligence Research*, vol. 57, pp. 229–271, 2016.

[97] T. Keller and P. Eyerich, "PROST: Probabilistic planning based on UCT," in *International Conference on Automated Planning and Scheduling*, 2012, pp. 119–127.

[98] E. Beaudry, F. Kabanza, and F. Michaud, "Planning with concurrency under resources and time uncertainty." in *European Conference on Artificial Intelligence*, vol. 2010, 2010, p. 217.

[99] A. J. Coles, "Opportunistic branched plans to maximise utility in the presence of resource uncertainty." in *European Conference on Artificial Intelligence*, vol. 2012, 2012, p. 252.

[100] I. Tsamardinos, "A probabilistic approach to robust execution of temporal plans with uncertainty," in *Hellenic Conference on Artificial Intelligence*, 2002, pp. 97–108.

[101] P. Santana, T. Vaquero, C. Toledo, A. Wang, C. Fang, and B. Williams, "Paris: A polynomial-time, risk-sensitive scheduling algorithm for probabilistic simple temporal networks with uncertainty," in *International Conference on Automated Planning and Scheduling*, 2016, pp. 267–275.

[102] K. Lund, S. Dietrich, S. Chow, and J. Boerkoel, “Robust execution of probabilistic temporal plans,” in *AAAI Conference on Artificial Intelligence*, 2017, pp. 3597–3604.

[103] P. Yu, C. Fang, and B. Williams, “Resolving over-constrained probabilistic temporal problems through chance constraint relaxation,” in *AAAI Conference on Artificial Intelligence*, 2015, pp. 3425–3431.

[104] A. Charnes, W. W. Cooper, and G. H. Symonds, “Cost horizons and certainty equivalents: an approach to stochastic programming of heating oil,” *Management Science*, vol. 4, no. 3, pp. 235–263, 1958.

[105] A. Charnes and W. W. Cooper, “Deterministic equivalents for optimizing and satisficing under chance constraints,” *Operations Research*, vol. 11, no. 1, pp. 18–39, 1963.

[106] A. Prékopa, “On probabilistic constrained programming,” in *Princeton Symposium on Mathematical Programming*, 1970, pp. 113–138.

[107] ——, “On logarithmic concave measures and functions,” 1973.

[108] ——, “Logarithmic concave measures with applications to stochastic programming,” 1971.

[109] A. Prekopa, “Contributions to the theory of stochastic programming,” *Mathematical Programming*, vol. 4, pp. 202–221, 1973.

[110] R. Henrion and C. Strugarek, “Convexity of chance constraints with independent random variables,” *Computational Optimization and Applications*, vol. 41, no. 2, pp. 263–276, 2008.

[111] ——, “Convexity of chance constraints with dependent random variables: the use of copulae,” in *Stochastic Optimization Methods in Finance and*

Energy: New Financial Products and Energy Market Strategies. Springer, 2011, pp. 427–439.

[112] M. Lubin, D. Bienstock, and J. P. Vielma, "Two-sided linear chance constraints and extensions," *arXiv preprint arXiv:1507.01995*, 2015.

[113] M. Bagnoli and T. Bergstrom, "Log-concave probability and its applications," in *Rationality and Equilibrium: A Symposium in Honor of Marcel K. Richter.* Springer, 2006, pp. 217–241.

[114] G. Zoutendijk, *Methods of Feasible Directions*, 1960.

[115] A. Prékopa, S. Ganczer, I. Deák, and K. Patyi, "The stabil stochastic programming model and its experimental application to the electrical energy sector of the Hungarian economy," *Stochastic Programming*, pp. 369–385, 1980.

[116] J. Luedtke and S. Ahmed, "A sample approximation approach for optimization with probabilistic constraints," *SIAM Journal on Optimization*, vol. 19, no. 2, pp. 674–699, 2008.

[117] J. E. Kelley, Jr, "The cutting-plane method for solving convex programs," *Journal of the society for Industrial and Applied Mathematics*, vol. 8, no. 4, pp. 703–712, 1960.

[118] A. Prékopa, B. Vizvari, and T. Badics, "Programming under probabilistic constraint with discrete random variable," in *New trends in mathematical programming: homage to Steven Vajda.* Springer, 1998, pp. 235–255.

[119] F. Serrano, R. Schwarz, and A. Gleixner, "On the relation between the extended supporting hyperplane algorithm and Kelley's cutting plane algorithm," *Journal of Global Optimization*, vol. 78, no. 1, pp. 161–179, 2020.

[120] A. F. Veinott Jr, “The supporting hyperplane method for unimodal programming,” *Operations Research*, vol. 15, no. 1, pp. 147–152, 1967.

[121] T. Arnold, R. Henrion, A. Möller, and S. Vigerske, “A mixed-integer stochastic nonlinear optimization problem with joint probabilistic constraints,” *Pacific Journal of Optimization*, vol. 10, pp. 5–20, 2014.

[122] T. Szántai, “A computer code for solution of probabilistic-constrained stochastic programming problems,” *Numerical Techniques for Stochastic Optimization*, vol. 10, p. 229, 1988.

[123] A. Prékopa, “Dual method for the solution of a one-stage stochastic programming problem with random RHS obeying a discrete probability distribution,” *Zeitschrift für Operations Research*, vol. 34, pp. 441–461, 1990.

[124] D. Dentcheva, A. Prékopa, and A. Ruszczynski, “Concavity and efficient points of discrete distributions in probabilistic programming,” *Mathematical programming*, vol. 89, pp. 55–77, 2000.

[125] L. Ricciardi Celsi, “The dilemma of rapid AI advancements: Striking a balance between innovation and regulation by pursuing risk-aware value creation,” *Information*, vol. 14, no. 12, p. 645, 2023.

[126] B. Verweij, S. Ahmed, A. J. Kleywegt, G. Nemhauser, and A. Shapiro, “The sample average approximation method applied to stochastic routing problems: a computational study,” *Computational optimization and applications*, vol. 24, pp. 289–333, 2003.

[127] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali, *Linear programming and network flows*. John Wiley & Sons, 2011.

[128] A. J. Conejo, E. Castillo, R. Minguez, and R. Garcia-Bertrand, *Decomposition techniques in mathematical programming: engineering and science applications.* Springer Science & Business Media, 2006.

[129] M. E. Lübbecke and J. Desrosiers, "Selected topics in column generation," *Operations research*, vol. 53, no. 6, pp. 1007–1023, 2005.

[130] G. B. Dantzig, A. Orden, P. Wolfe *et al.*, "The generalized simplex method for minimizing a linear form under linear inequality restraints," *Pacific Journal of Mathematics*, vol. 5, no. 2, pp. 183–195, 1955.

[131] V. Chvatal, *Linear programming.* Macmillan, 1983.

[132] G. Desaulniers, J. Desrosiers, and M. M. Solomon, *Column generation.* Springer Science & Business Media, 2006, vol. 5.

[133] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. Savelsbergh, and P. H. Vance, "Branch-and-price: Column generation for solving huge integer programs," *Operations research*, vol. 46, no. 3, pp. 316–329, 1998.

[134] L. V. Kantorovich, "Mathematical methods of organizing and planning production," *Management science*, vol. 6, no. 4, pp. 366–422, 1960.

[135] J. Sessions, E. Olsen, and J. Garland, "Tree bucking for optimal stand value with log allocation constraints," *Forest Science*, vol. 35, no. 1, pp. 271–276, 1989.

[136] G. A. Ogunranti and A. E. Oluleye, "Minimizing waste (off-cuts) using cutting stock model: The case of one dimensional cutting stock problem in wood working industry," *Journal of Industrial Engineering and Management (JIEM)*, vol. 9, no. 3, pp. 834–859, 2016.

[137] C. Kazunga, L. Mutambara, and J. Mapurisa, "A column generation approach to a carpentry cutting stock problem: a case study for planks cutting

in Zimbabwe," *African Journal of Educational Studies in Mathematics and Sciences*, vol. 9, no. 1, pp. 49–59, 2011.

[138] C. Goulimis, "Optimal solutions for the cutting stock problem," *European Journal of Operational Research*, vol. 44, no. 2, pp. 197–208, 1990.

[139] F. K. Lemos, A. C. Cherri, and S. A. de Araujo, "The cutting stock problem with multiple manufacturing modes applied to a construction industry," *International Journal of Production Research*, vol. 59, no. 4, pp. 1088–1106, 2021.

[140] S. L. Nonås and A. Thorstenson, "A combined cutting-stock and lot-sizing problem," *European Journal of Operational Research*, vol. 120, no. 2, pp. 327–342, 2000.

[141] S. Martello and P. Toth, *Knapsack problems: algorithms and computer implementations.* John Wiley & Sons, Inc., 1990.

[142] M. Johnson, C. Rennick, and E. Zak, "Case studies from industry: skiving addition to the cutting stock problem in the paper industry," *SIAM Review*, vol. 39, no. 3, pp. 472–483, 1997.

[143] F. Vanderbeck, "Computational study of a column generation algorithm for bin packing and cutting stock problems," *Mathematical Programming*, vol. 86, pp. 565–594, 1999.

[144] H. Yang, T. So, and Y. Xu, "5G network slicing," in *5G NR and Enhancements.* Elsevier, 2022, pp. 621–639.

[145] ETSI, "Network functions virtualisation (NFV) architectural framework," Tech. Rep. NFV-002, 2014.

[146] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, “Network function virtualization: Challenges and opportunities for innovations,” *IEEE Communications Magazine*, vol. 53, no. 2, pp. 90–97, 2015.

[147] B. Addis, D. Belabed, M. Bouet, and S. Secci, “Virtual network functions placement and routing optimization,” in *IEEE International Conference on Cloud Networking.* IEEE, 2015, pp. 171–177.

[148] H. Moens and F. De Turck, “VNF-P: A model for efficient placement of virtualized network functions,” in *International Conference on Network and Service Management*, 2014, pp. 418–423.

[149] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, “On orchestrating virtual network functions,” in *International Conference on Network and Service Management*, 2015, pp. 50–56.

[150] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary, “Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions,” in *IFIP/IEEE International Symposium on Integrated Network Management*, 2015, pp. 98–106.

[151] D. Bhamare, M. Samaka, A. Erbad, R. Jain, L. Gupta, and H. A. Chan, “Optimal virtual network function placement in multi-cloud service function chaining architecture,” *Computer Communications*, vol. 102, pp. 1–16, 2017.

[152] A. Laghrissi, T. Taleb, M. Bagaa, and H. Flinck, “Towards edge slicing: VNF placement algorithms for a dynamic & realistic edge cloud environment,” in *IEEE Global Communications Conference*, 2017, pp. 1–6.

[153] L. Yala, P. A. Frangoudis, G. Lucarelli, and A. Ksentini, “Cost and availability aware resource allocation and virtual function placement for CD-

NaaS provision," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1334–1348, 2018.

[154] F. Carpio and A. Jukan, "Improving reliability of service function chains with combined VNF migrations and replications," *arXiv preprint arXiv:1711.08965*, 2017.

[155] P. Vizarreta, M. Condoluci, C. M. Machuca, T. Mahmoodi, and W. Kellerer, "QoS-driven function placement reducing expenditures in NFV deployments," in *IEEE International Conference on Communications*, 2017, pp. 1–7.

[156] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "SNDlib 1.0—survivable network design library," *Networks: An International Journal*, vol. 55, no. 3, pp. 276–286, 2010.

[157] S. Haller, "The things in the internet of things," *Poster at the (IoT 2010). Tokyo, Japan, November*, vol. 5, no. 8, pp. 26–30, 2010.

[158] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE communications surveys & tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[159] M. Series, "IMT vision–framework and overall objectives of the future development of IMT for 2020 and beyond," *Recommendation ITU*, vol. 2083, no. 0, pp. 11–12, 2015.

[160] C-RAN, "C-RAN: The road towards green RAN," Tech. Rep. 2.5, 2011.

[161] J. Wu, Z. Zhang, Y. Hong, and Y. Wen, "Cloud radio access network (C-RAN): a primer," *IEEE Network*, vol. 29, no. 1, pp. 35–41, 2015.

[162] L. I. Barona López, Á. L. Valdivieso Caraguay, L. J. García Villalba, and D. López, “Trends on virtualisation with software defined networking and network function virtualisation,” *IET Networks*, vol. 4, no. 5, pp. 255–263, 2015.

[163] J. G. Herrera and J. F. Botero, “Resource allocation in NFV: A comprehensive survey,” *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 518–532, 2016.

[164] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, “Network function virtualization: State-of-the-art and research challenges,” *IEEE Communications surveys & tutorials*, vol. 18, no. 1, pp. 236–262, 2015.

[165] C. Cui, H. Deng, D. Telekom, U. Michel, and H. Damker, “Network functions virtualisation,” 2012.

[166] J. Martins, M. Ahmed, C. Raiciu, V. Olteanu, M. Honda, R. Bifulco, and F. Huici, “{ClickOS} and the art of network function virtualization,” in *Symposium on Networked Systems Design and Implementation*, 2014, pp. 459–473.

[167] W. Attaoui, E. Sabir, H. Elbiaze, and M. Guizani, “VNF and CNF placement in 5G: Recent advances and future trends,” *IEEE Transactions on Network and Service Management*, vol. 20, pp. 4698–4733, 2023.

[168] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, “A break in the clouds: towards a cloud definition,” *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 50–55, 2008.

[169] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog computing and its role in the internet of things,” in *MCC workshop on Mobile cloud computing*, 2012, pp. 13–16.

[170] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge computing: Vision and challenges,” *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[171] A. Laghrissi and T. Taleb, “A survey on the placement of virtual resources and virtual network functions,” *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1409–1434, 2018.

[172] J. Sun, Y. Zhang, F. Liu, H. Wang, X. Xu, and Y. Li, “A survey on the placement of virtual network functions,” *Journal of Network and Computer Applications*, vol. 202, p. 103361, 2022.

[173] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, “Near optimal placement of virtual network functions,” in *IEEE Conference on Computer Communications*, 2015, pp. 1346–1354.

[174] A. Fischer, J. F. Botero, M. T. Beck, H. De Meer, and X. Hesselbach, “Virtual network embedding: A survey,” *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 1888–1906, 2013.

[175] S. Mehraghdam, M. Keller, and H. Karl, “Specifying and placing chains of virtual network functions,” in *IEEE International Conference on Cloud Networking*, 2014, pp. 7–13.

[176] F. Carpio, S. Dhahri, and A. Jukan, “VNF placement with replication for loac balancing in NFV networks,” in *IEEE International Conference on Communications*, 2017, pp. 1–6.

[177] M. Golkarifard, C. F. Chiasserini, F. Malandrino, and A. Movaghar, "Dynamic vnf placement, resource allocation and traffic routing in 5g," *Computer Networks*, vol. 188, p. 107830, 2021.

[178] J. Liu, W. Lu, F. Zhou, P. Lu, and Z. Zhu, "On dynamic service function chain deployment and readjustment," *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 543–553, 2017.

[179] A. Baumgartner, V. S. Reddy, and T. Bauschert, "Mobile core network virtualization: A model for combined virtual core network function placement and topology optimization," in *IEEE Conference on Network Softwarization*, 2015, pp. 1–9.

[180] A. Mohammadkhan, S. Ghapani, G. Liu, W. Zhang, K. Ramakrishnan, and T. Wood, "Virtual function placement and traffic steering in flexible and dynamic software defined networks," in *IEEE International Workshop on Local and Metropolitan Area Networks*, 2015, pp. 1–6.

[181] R. Riggio, A. Bradai, T. Rasheed, J. Schulz-Zander, S. Kuklinski, and T. Ahmed, "Virtual network functions orchestration in wireless networks," in *International Conference on Network and Service Management*, 2015, pp. 108–116.

[182] F. B. Jemaa, G. Pujolle, and M. Pariente, "QoS-aware VNF placement optimization in edge-central carrier cloud architecture," in *IEEE Global Communications Conference*, 2016, pp. 1–7.

[183] M. Ghaznavi, N. Shahriar, R. Ahmed, and R. Boutaba, "Service function chaining simplified," *arXiv preprint arXiv:1601.00751*, 2016.

[184] A. Hmaity, M. Savi, F. Musumeci, M. Tornatore, and A. Pattavina, "Virtual network function placement for resilient service chain provisioning," in

International Workshop on Resilient Networks Design and Modeling, 2016, pp. 245–252.

[185] Q. Sun, P. Lu, W. Lu, and Z. Zhu, “Forecast-assisted NFV service chain deployment based on affiliation-aware VNF placement,” in *IEEE Global Communications Conference.* IEEE, 2016, pp. 1–6.

[186] A. Gupta, M. F. Habib, U. Mandal, P. Chowdhury, M. Tornatore, and B. Mukherjee, “On service-chaining strategies using virtual network functions in operator networks,” *Computer Networks*, vol. 133, pp. 1–16, 2018.

[187] A. Ali, C. Anagnostopoulos, and D. P. Pezaros, “On the optimality of virtualized security function placement in multi-tenant data centers,” in *IEEE International Conference on Communications*, 2018, pp. 1–6.

[188] G. B. Dantzig and J. H. Ramser, “The truck dispatching problem,” *Management science*, vol. 6, no. 1, pp. 80–91, 1959.

[189] M. L. Balinski and R. E. Quandt, “On an integer program for a delivery problem,” *Operations research*, vol. 12, no. 2, pp. 300–304, 1964.

[190] N. Huin, B. Jaumard, and F. Giroire, “Optimal network service chain provisioning,” *IEEE/ACM Transactions on Networking*, vol. 26, no. 3, pp. 1320–1333, 2018.

[191] J. Cao, Y. Zhang, W. An, X. Chen, J. Sun, and Y. Han, “VNF-FG design and VNF placement for 5G mobile networks,” *Science China Information Sciences*, vol. 60, pp. 1–15, 2017.

[192] S. Agarwal, F. Malandrino, C.-F. Chiasserini, and S. De, “Joint VNF placement and CPU allocation in 5G,” in *IEEE Conference on Computer Communications*, 2018, pp. 1943–1951.

[193] P. M. Mohan and M. Gurusamy, “Resilient VNF placement for service chain embedding in diversified 5G network slices,” in *IEEE Global Communications Conference*, 2019, pp. 1–6.

[194] S. Yang, F. Li, S. Trajanovski, X. Chen, Y. Wang, and X. Fu, “Delay-aware virtual network function placement and routing in edge clouds,” *IEEE Transactions on Mobile Computing*, vol. 20, no. 2, pp. 445–459, 2019.

[195] A. Varasteh, B. Madiwalar, A. Van Bemten, W. Kellerer, and C. Mas-Machuca, “Holu: Power-aware and delay-constrained VNF placement and chaining,” *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1524–1539, 2021.

[196] H. Liu, F. Eldarrat, H. Alqahtani, A. Reznik, X. De Foy, and Y. Zhang, “Mobile edge cloud system: Architectures, challenges, and approaches,” *IEEE Systems Journal*, vol. 12, no. 3, pp. 2495–2508, 2017.

[197] D. Li, P. Hong, K. Xue, and J. Pei, “Availability aware VNF deployment in datacenter through shared redundancy and multi-tenancy,” *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1651–1664, 2019.

[198] Y. Alahmad, A. Agarwal, and T. Daradkeh, “Cost and availability-aware VNF selection and placement for network services in NFV,” in *International Symposium on Networks, Computers and Communications*, 2020, pp. 1–6.

[199] S. Yang, S. Trajanovski, and F. A. Kuipers, “Availability-based path selection,” in *International Workshop on Reliable Networks Design and Modeling*, 2014, pp. 39–46.

[200] A. Goldberg and T. Radzik, “A heuristic improvement of the Bellman-Ford algorithm,” Stanford University Department of Computer Science, Tech. Rep., 1993.

[201] N. Promwongsa, A. Ebrahimzadeh, R. H. Glitho, and N. Crespi, "Joint VNF placement and scheduling for latency-sensitive services," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 4, pp. 2432–2449, 2022.

[202] S. M. Araujo, F. S. de Souza, and G. R. Mateus, "A demand aware strategy for a machine learning approach to VNF-PC problem," in *IEEE International Conference on Cloud Networking*, 2022, pp. 211–219.

[203] Y. Gu, Y. Hu, Y. Ding, J. Lu, and J. Xie, "Elastic virtual network function orchestration policy based on workload prediction," *IEEE Access*, vol. 7, pp. 96 868–96 878, 2019.

[204] J. Cai, K. Qian, J. Luo, and K. Zhu, "SARM: service function chain active reconfiguration mechanism based on load and demand prediction," *International Journal of Intelligent Systems*, vol. 37, no. 9, pp. 6388–6414, 2022.

[205] M. Wang, B. Cheng, and J. Chen, "Joint availability guarantee and resource optimization of virtual network function placement in data center networks," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 821–834, 2020.

[206] A. Jalalian, S. Yousefi, and T. Kunz, "Network slicing in virtualized 5G core with VNF sharing," *Journal of Network and Computer Applications*, p. 103631, 2023.

[207] 3GPP, "Service requirements for cyber-physical control applications in vertical domains, rel. 16," Tech. Rep. 22.104, 2018.

[208] Cisco, "Cisco annual internet report (2018–2023) white paper," Tech. Rep., 2020.

[209] Gurobi, "Gurobi 8 performance benchmarks," https://assets.gurobi.com/pdfs/benchmarks.pdf, 2019, [Online; accessed 17-May-2024].

[210] T. Achterberg, "SCIP: solving constraint integer programs," *Mathematical Programming Computation*, vol. 1, pp. 1–41, 2009.

[211] S. Maher, M. Miltenberger, J. P. Pedroso, D. Rehfeldt, R. Schwarz, and F. Serrano, "Pyscipopt: Mathematical programming in python with the scip optimization suite," in *International Conference on Mathematical Software*. Springer, 2016, pp. 301–307.

[212] T. Vidal and M. Ghallab, "Dealing with uncertain durations in temporal constraint networks dedicated to planning," in *European Conference on Artificial Intelligence*, 1996, pp. 48–54.

[213] T. Vidal and H. Fargier, "Handling contingency in temporal constraint networks: From consistency to controllabilities," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 11, pp. 23–45, 01 1999.

[214] G. Filippi, M. Vasile, E. Patelli, and M. Fossati, "Generative optimisation of resilient drone logistic networks," in *IEEE Congress on Evolutionary Computation*, 2022.

[215] I. Bakach, A. M. Campbell, J. F. Ehmke, and T. L. Urban, "Solving vehicle routing problems with stochastic and correlated travel times and makespan objectives," *EURO Journal on Transportation and Logistics*, vol. 10, p. 100029, 2021.

[216] D. Park and L. R. Rilett, "Forecasting freeway link travel times with a multilayer feedforward neural network," *Computer-Aided Civil and Infrastructure Engineering*, vol. 14, no. 5, pp. 357–367, 1999.

[217] J. Gondzio, P. González-Brevis, and P. Munari, “Large-scale optimization with the primal-dual column generation method,” *Mathematical Programming Computation*, vol. 8, no. 1, pp. 47–82, 2016.

[218] P. H. Morris and N. Muscettola, “Temporal dynamic controllability revisited,” in *AAAI Conference on Artificial Intelligence*, 2005, pp. 1193–1198.

[219] A. Cimatti, A. Micheli, and M. Roveri, “Solving strong controllability of temporal problems with uncertainty using SMT,” *Constraints*, vol. 20, pp. 1–29, 2015.

[220] T. Vidal and H. Fargier, “Contingent durations in temporal CSPs: from consistency to controllabilities,” in *International Workshop on Temporal Representation and Reasoning*, 1997, pp. 78–85.

[221] A. Wang and B. Williams, “Chance-constrained scheduling via conflict-directed risk allocation,” in *AAAI Conference on Artificial Intelligence*, vol. 29, no. 1, 2015, pp. 3620–3627.

[222] C.-C. Lu, K.-C. Ying, and S.-W. Lin, “Robust single machine scheduling for minimizing total flow time in the presence of uncertain processing times,” *Computers & Industrial Engineering*, vol. 74, pp. 102–110, 2014.

[223] A. Moreira, A. Street, and J. M. Arroyo, “Energy and reserve scheduling under correlated nodal demand uncertainty: An adjustable robust optimization approach,” *International Journal of Electrical Power & Energy Systems*, vol. 72, pp. 91–98, 2015.

[224] Ö. Ökmen and A. Öztaş, “Construction project network evaluation with correlated schedule risk analysis model,” *Journal of Construction Engineering and Management*, vol. 134, no. 1, pp. 49–63, 2008.

[225] W.-C. Wang and L. A. Demsetz, “Model for evaluating networks under correlated uncertainty—NETCOR,” *Journal of Construction Engineering and Management*, vol. 126, no. 6, pp. 458–466, 2000.

[226] G. Maronati and B. Petrovic, “Estimating cost uncertainties in nuclear power plant construction through monte carlo sampled correlated random variables,” *Progress in Nuclear Energy*, vol. 111, pp. 211–222, 2019.

[227] R. Eiris Pereira and I. Flood, “Impact of linear correlation on construction project performance using stochastic linear scheduling,” *Visualization in Engineering*, vol. 5, no. 1, pp. 1–12, 2017.

[228] A. Nicholson, “Travel time reliability benefits: Allowing for correlation,” *Research in Transportation Economics*, vol. 49, pp. 14–21, 2015.

[229] R. Henrion and A. Möller, “A gradient formula for linear chance constraints under gaussian distribution,” *Mathematics of Operations Research*, vol. 37, no. 3, pp. 475–488, 2012.

[230] A. Saumard and J. A. Wellner, “Log-concavity and strong log-concavity: a review,” *Statistics surveys*, vol. 8, p. 45, 2014.

[231] W. Van Ackooij, “A discussion of probability functions and constraints from a variational perspective,” *Set-Valued and Variational Analysis*, vol. 28, no. 4, pp. 585–609, 2020.

[232] C. I. Fábián, “Gaining traction: on the convergence of an inner approximation scheme for probability maximization,” *Central European Journal of Operations Research*, vol. 29, no. 2, pp. 491–519, 2021.

[233] A. M. Geoffrion, “Elements of large-scale mathematical programming part I: concepts,” *Management Science*, vol. 16, no. 11, p. 652–675, 1970.

[234] G. B. Dantzig, *Linear Programming and Extensions.* Princeton: Princeton University Press, 1963.

[235] A. Genz, “Numerical computation of multivariate normal probabilities,” *Journal of Computational and Graphical Statistics*, vol. 1, no. 2, pp. 141–149, 1992.

[236] W. Van Ackooij, R. Henrion, A. Möller, and R. Zorgati, “On probabilistic constraints induced by rectangular sets and multivariate normal distributions,” *Mathematical Methods of Operations Research*, vol. 71, no. 3, pp. 535–549, 2010.

[237] W. Van Ackooij, R. Zorgati, R. Henrion, and A. Möller, “Chance constrained programming and its applications to energy management,” *Stochastic Optimization-Seeing the Optimal for the Uncertain*, pp. 291–320, 2011.

[238] M. Fox and D. Long, “PDDL2. 1: An extension to PDDL for expressing temporal planning domains,” *Journal of Artificial Intelligence Research*, vol. 20, pp. 61–124, 2003.

[239] J. Benton, A. Coles, and A. Coles, “Temporal planning with preferences and time-dependent continuous costs,” in *International Conference on Automated Planning and Scheduling*, vol. 22, 2012, pp. 2–10.

[240] A. Coles, A. Coles, A. G. Olaya, S. Jiménez, C. L. López, S. Sanner, and S. Yoon, “A survey of the seventh international planning competition,” *Ai Magazine*, vol. 33, no. 1, pp. 83–88, 2012.

[241] J. Desrosiers and M. E. Lübbecke, “A primer in column generation,” in *Column Generation.* Springer, 2005, pp. 1–32.

[242] T. G. Dietterich, “Steps toward robust artificial intelligence,” *AI Magazine*, vol. 38, no. 3, pp. 3–24, 2017.

[243] E. Zemler, S. Azimi, K. Chang, J. Frank, and R. Morris, “Integrating task planning with robust execution for autonomous robotic manipulation in space,” in *ICAPS Workshop on Planning and Robotics*, vol. 3, 2020.

[244] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma, “A survey on ensemble learning,” *Frontiers of Computer Science*, vol. 14, pp. 241–258, 2020.

[245] A. Fern and P. Lewis, “Ensemble Monte-Carlo planning: An empirical study,” in *International Conference on Automated Planning and Scheduling*, vol. 21, 2011, pp. 58–65.

[246] S. Ghosh, S. Laguna, S. H. Lim, L. Wynter, and H. Poonawala, “A deep ensemble method for multi-agent reinforcement learning: A case study on air traffic control,” in *International Conference on Automated Planning and Scheduling*, vol. 31, 2021, pp. 468–476.

[247] M. Shen and J. P. How, “Robust opponent modeling via adversarial ensemble reinforcement learning,” in *International Conference on Automated Planning and Scheduling*, vol. 31, 2021, pp. 578–587.

[248] A. Ruszczyński and A. Świtanowski, “Accelerating the regularized decomposition method for two stage stochastic linear problems,” *European Journal of Operational Research*, vol. 101, no. 2, pp. 328–342, 1997.

[249] M. Lee, N. Ma, G. Yu, and H. Dai, “Accelerating generalized Benders decomposition for wireless resource allocation,” *IEEE Transactions on Wireless Communications*, vol. 20, no. 2, pp. 1233–1247, 2020.

[250] K. Mana, S. Mak, P. Zehtabi, M. Cashmore, D. Magazzeni, and M. Veloso, “Accelerating Benders decomposition via reinforcement learning surrogate

models," *ICAPS Workshop on Planning and Scheduling for Financial Services*, p. 12, 2023.

[251] N. Yu, B. Qian, R. Hu, Y. Chen, and L. Wang, "Solving open vehicle problem with time window by hybrid column generation algorithm," *Journal of Systems Engineering and Electronics*, vol. 33, no. 4, pp. 997–1009, 2022.

[252] S. Kraul, M. Seizinger, and J. O. Brunner, "Machine learning–supported prediction of dual variables for the cutting stock problem with an application in stabilized column generation," *INFORMS Journal on Computing*, 2023.

[253] F. Nawaz, O. Hussain, F. K. Hussain, N. K. Janjua, M. Saberi, and E. Chang, "Proactive management of SLA violations by capturing relevant external events in a cloud of things environment," *Future Generation Computer Systems*, vol. 95, pp. 26–44, 2019.

[254] F. Nawaz, N. K. Janjua, O. K. Hussain, F. K. Hussain, E. Chang, and M. Saberi, "Event-driven approach for predictive and proactive management of SLA violations in the cloud of things," *Future Generation Computer Systems*, vol. 84, pp. 78–97, 2018.

[255] M. Gao, L. Popowski, and J. Boerkoel, "Dynamic control of probabilistic simple temporal networks," in *AAAI Conference on Artificial Intelligence*, vol. 34, no. 06, 2020, pp. 9851–9858.

[256] M. Saint-Guillain, T. S. Vaquero, J. Agrawal, and S. Chien, "Robustness computation of dynamic controllability in probabilistic temporal networks with ordinary distributions," in *International Joint Conferences on Artificial Intelligence*, 2021, pp. 4168–4175.

[257] P. Laborie, "Resource temporal networks: Definition and complexity," in *International Joint Conference on Artificial Intelligence*, 2003, pp. 948–953.

[258] C. Combi, R. Posenato, L. Viganò, and M. Zavatteri, "Conditional simple temporal networks with uncertainty and resources," *Journal of Artificial Intelligence Research*, vol. 64, pp. 931–985, 2019.

[259] M. Cashmore, A. Cimatti, D. Magazzeni, A. Micheli, and P. Zehtabi, "Robustness envelopes for temporal plans," in *AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 7538–7545.