Optimal decision-making under constraints and uncertainty

Latour, A.L.D.

1

Introduction

> This is where we've filled ourselves up with so many questions that they're starting to overflow and become answers.
>
> Sir Terry Pratchett

1.1 Motivation

In business, governance, science as well as in our daily lives, we often have to solve problems that involve decision making under constraints and uncertainty. Examples of these problems arise in a diversity of domains, such as:

- planning (*e.g.*, decide when to invest in which company or product, to maximise your return on investment) [7],

- scheduling (*e.g.*, finding rosters for nurses that honour their preferences as much as possible and can deal with a stochastic number of daily patients) [181],
- production planning (*e.g.*, deciding how many books to print in each quarter of the year, to minimise storage costs but have enough to satisfy customers) [181],
- vehicle routing (*e.g.*, deliver all the packages ordered by people who are under sheltering-in-place during a pandemic as efficiently as possible, while dealing with stochastic demand) [150], and even
- bioinformatics (*e.g.*, model stochastic protein-protein and protein-gene interaction in a network that is as small as possible, but still explains the interactions as true to nature as possible) [50, 133].

Note that all the above examples do not just require us to make a decision under constraints and uncertainty, but also require some kind of *optimality* of that decision. We want to maximise profit or minimise cost. We want to maximise efficiency or minimise size. This is a common property of decision making problems in the world around us. In real-world problems, we often face multiple, possibly conflicting, objectives, such that solutions to a problem contain a certain trade-off with respect to these objectives. These trade-offs can often be captured in a single cost or utility value. We therefore only consider single-objective optimisation problems in this work.

Given the abundance of relational data in the areas mentioned above, many problems also involve probabilistic network data [50, 56, 61, 92]. Consider, for example, the following two problems.

Spread of influence This is a problem setting that is well-known from the data mining literature [56, 92]. We are given a social network in which the nodes represent people and the edges represent probabilistic mutual influence relationships, such as people following each other on social media. We are also given a marketing budget, which we can spend on providing free samples of our product to selected individuals in the social network. We then rely on a word-of-mouth marketing strategy, in which people who have tried our product may become our customers and try to convince their friends, family and acquaintances to also buy the product and become a customer. Depending on how much influence they hold over those relations, they are either successful in convincing a relation, or not. Our objective is to maximise the number of people in the social network who are convinced that they should buy our product, and are thus converted to customers. We can start this process by using our budget to provide free samples to a subset of the people in the network, which we distribute all at the same time. We

must therefore identify the subset of people that is small enough such that we do not exceed our budget (constraint), but maximises the expected number of people who will eventually be convinced to buy our product (optimisation criterion).

Power grid reliability This is an optimisation version of a problem known from the literature [61]. We are given a high-voltage power grid in which the nodes may either represent power producers (like nuclear power plants or solar farms), power consumers (power stations that transform the high-voltage power into lower-voltage power to distribute among buildings), and power transmitters (power stations that simply pass on the power, possibly splitting or merging lines). Power lines connect nodes to each other. With each power line we associate a probability that it will remain intact during a natural disaster like an earthquake, hurricane or storm surge. This probability may depend on its length or the terrain in which it exists. If too many lines are damaged, consumers may lose power. We are given a power line maintenance budget (constraint) and must decide which power lines we spend it on, such that we maximise the expected number of power consumers that are still connected to at least one power producer after a natural disaster (optimisation criterion). The budget must be assigned at a single moment in time.

Note that, because for each problem setting, we have to decide how to spend the budget in one moment in time, both these problem settings are considered *single-stage* constraint optimisation problems. These problems are instances of a general class of problems, known as *stochastic constraint (optimisation) problems (SCPs)*. SCPs have the following characteristics:

- They involve (Boolean) *decision* variables and (Boolean) *stochastic* (or *random*) variables.
- They involve reasoning over probability distributions.
- They involve constraints that limit the decisions we can make.
- They involve an optimisation criterion.

We provide a formal description of SCPs in Section 1.2.

For this work, we have chosen to limit ourselves to studying single-stage problems. As there are many real-world single-state problems, some of which were mentioned above, we believe that this choice to limit the scope is justified. We study how to solve SCPs efficiently, and use optimisations that are only possible in the single-stage setting.

The main goal of this work is to develop methods that find exact solutions to single-stage SCPs that are formulated on probabilistic networks, making sure that these methods strike a reasonable balance between:

Convenience, such that our methods and tools are accessible and easy to use, even for people with little or no background in programming or computer science.

Generality, such that our methods and tools can be used for solving a diversity of problem and problem types, from different application domains, and

Speed, such that our methods and tools solve SCPs fast enough to be practical.

In meeting the last requirement, we attempt to find algorithms with a low theoretical bound on the running time, as well as algorithms that have the potential to be faster in practice than others, even if their theoretical complexity is not lower. In order to find these exact solutions, we need to take three distinct steps:

Model the problem mathematically: Define the real-world problem and what constitutes a solution to that problem.

Specify the model in a computer-friendly manner: Use a suitable programming language to communicate the problem to a computer.

Solve the problem: Let an algorithm find the exact optimal solution.

In the literature, this second step is also referred to as 'modelling', so we will use that term in the remainder of this dissertation for different tasks, trusting that the context is enough to disambiguate, and clarifying wherever necessary.

The remainder of this chapter is organised as follows. In Section 1.2, we describe the stochastic constraint that is central to this work. Then, in Section 1.3 we briefly reflect on the hardness of SCPs and how we address the computational complexity of SCPs. We list and motivate our main research questions in Section 1.4, and specify which contributions we present in this dissertation in relation to these questions. This last section also serves as an outline to the remainder of this dissertation.

1.2 Stochastic constraints on probability distributions

The SCPs that we aim to solve in this work are all characterised by the presence of a stochastic constraint, similar to the ones studied by Papadimitriou [137] and

Littman *et al.* [111], or a stochastic optimisation criterion, similar to the ones studied by Walsh [181] and Van den Broeck *et al.* [178]. Specifically, in this work we study stochastic constraints of the following form:

$$\sum_{\phi \in \Phi} \rho_\phi \cdot P\left(\phi \mid \sigma\right) > \theta. \tag{1.1}$$

The sum represents an *expected utility*, in which Φ is a set of stochastic events that are of interest to us, $P\left(\phi \mid \sigma\right)$ represents the probability of an event ϕ happening, given a *strategy* σ over Boolean variables; and $\rho_\phi \in \mathbb{R}^+$ is a reward for this event. This constraint specifies a *lower bound* $\theta \in \mathbb{R}^+$ for an expected utility. Our methods can also be applied to stochastic constraints that impose an *upper bound* on an expected utility, instead.

Recall that SCPs are problems that may involve a stochastic *objective function*, rather than a stochastic *constraint*. We can straightforwardly employ constraints on probability distributions to solve minimisation or maximisation problems over expected utilities (under other constraints). We describe how this can be done in Chapter 3.

1.3 Computational complexity of SCPs

SCPs are difficult to solve exactly (*i.e.*, in a way that produces provably optimal solutions). Indeed, well-known instances of SCPs are shown to be $\mathcal{NP}$-hard [92], and solving SCPs exactly is $\mathcal{NP}$-hard in the general case. Specifically, exact SCP solving involves two components:

1. To evaluate the quality of a strategy σ, we have to compute $P(\phi \mid \sigma)$, which involves a counting task that is $\#\mathcal{P}$-complete in general [155, 176, 177].
2. We have to perform this evaluation a potentially exponential number of times, since the number of possible strategies for $|\mathbf{D}|$ Boolean decision variables is $2^{|\mathbf{D}|}$.

Informally, a $\#\mathcal{P}$-complete problem requires the counting of all solutions to a Boolean formula, of which it may have exponentially many, and can be harder than determining if a propositional formula has a solution, which is $\mathcal{NP}$-complete [38]. Thus, naïvely solving an SCP by enumerating all possible strategies and evaluating their score, which requires counting all the possible consequences of a strategy, is typically computationally impractical. We discuss $\mathcal{NP}$, $\#\mathcal{P}$ and other relevant complexity classes, as well as propositional formulae and the counting task referred to above, in Section 2.2.

There is earlier work on which we can build to address the two challenges listed above. In particular, *knowledge compilation* [48, 123, 165] techniques have been used to make probabilistic inference tractable. Similarly, *constraint programming (CP)* [154] or *mixed integer programming (MIP)* [25] have been used to model and solve constraint optimisation problems. In this work, we investigate whether and how these approaches can be combined to solve SCPs quickly, in theory or in practice.

The answer to that question is not immediately obvious. In order to reap the benefits of knowledge compilation, we have to encode the resulting representations of probability distributions in such a way that they can be communicated to a CP or MIP solver, otherwise the associated stochastic constraints cannot be solved by these solvers. We then have to choose how to model these constraints such that they can not only be solved quickly, but are also convenient and easy for the user to specify, and ideally generic enough to be implemented in a range of different solvers. The focus of this dissertation is on finding SCP solving methods with reasonable trade-offs between convenience, generality and speed.

1.4 Contributions

In this work we aim to answer four main research questions. We start by acknowledging that a new technology's success stands or falls on accessibility and ease-of-use. We therefore ask:

MRQ1 How can we conveniently model SCPs and specify them to a computer?

The contributions of this thesis with respect to **MRQ1** are as follows:

C1 We formulate a *stochastic constraint on probability distributions (SCPD)*, which allows us to model SCPs. As we will show in Section 4.2, this constraint can also be used to formulate stochastic optimisation problems.

C2 Next, we develop a new declarative programming language, *stochastic constraint probabilistic Prolog*, or SC-ProbLog, for programming SCPs. This language builds on earlier logic programming languages that allow for convenient modelling of probability distributions, and extends these with syntax and semantics for modelling constraints and optimisation criteria.

As we described in Section 1.3, it is not immediately obvious how we can use existing CP, MIP and knowledge compilation techniques to solve SCPs. We therefore ask:

MRQ2 How can we leverage CP, MIP and knowledge compilation technology to solve SCPs?

The contributions of this thesis with respect to **MRQ2** are as follows:

C3 We develop an SCPs solving pipeline, which takes as input an SCPs programmed in SC-ProbLog. It grounds the program and converts the resulting logic formulae into either *ordered binary decision diagrams (OBDDs)* or *sentential decision diagrams (SDDs)*. We either impose a stochastic constraint on the *decision diagram (DD)* representations of these probability distributions, or formulate an optimisation criterion that aims to maximise or minimise an expected utility that is computed from these probability distributions. We then *decompose* the OBDD or SDD into a CP or MIP model, which we solve using off-the-shelf solvers.

An important part of this work focuses on how encodings of probability distributions that are obtained through knowledge compilation interact with the CP and MIP solvers that we use to solve the SCPs that are formulated on those probability distributions. That part of this work is done to answer the following research question:

MRQ3 How can we leverage the properties of SDDs and OBDDs for faster SCP solving?

The contributions of this dissertation with respect to **MRQ3** are as follows:

C4 We observe that MIPs are much easier to solve if they are linear, rather than quadratic, and decomposed SDDs typically do not yield linear MIPs. Additionally, we observe that smaller SDDs yield smaller MIPs and show that smaller MIPs tend to take less long to solve than larger MIPs. To address these observations, we identify a class of SDDs that yield linear MIP decompositions and develop a minimisation algorithm for finding minimised SDDs that belong to this subset of SDDs.

C5 We show that CPs solvers cannot guarantee *generalised arc consistency (GAC)* in a naïve decomposition of stochastic constraints on OBDD representations of probability distributions. This results in the CP solver potentially searching a part of the search space that does not contain any feasible solutions, and thus wasting computation time. We also show that a GAC-guaranteeing decomposition of such a constraint comes at the cost of extra memory use and does not improve solving times significantly.

C6 To remedy these shortcomings, we introduce a novel, *global* constraint on probability distributions that are represented by OBDDs and have a certain *monotonic* property. We also present and implement a propagation algorithm for this *stochastic constraint on monotonic distributions (SCMD)*. This propagator leverages the structure of the OBDDs to incrementally compute the solution to the constraint, and the fact that the underlying probability distribution is monotonic to guarantee GAC.

In addressing **MRQ1** to **MRQ3**, we develop a number of different SCP solving methods, each with a number of different components, that can each be implemented in different ways. Since SCPs are hard to solve in general, and since each application domain may yield SCPs with different properties, it is not immediately obvious which solving method with which exact implementation choices for its different components works well to solve SCPs from a specific application domain. Additionally, since we use different existing tools (such as CP solvers Gecode[1] and OscaR [132], MIP solver Gurobi[2], and knowledge compilers CUDD [168] and sdd [36]), whose default parameter settings may be tuned on use cases that are rather different from the one studied in this work, it is unclear what a good parameter setting for these components might be. These uncertainties form a challenge for any scientist (or other user) who not only wants to evaluate the performance of these methods in a fair and informative manner, but also wants to use them as effectively as possible. This observation naturally begs an additional research question for this dissertation:

MRQ4 How can we fairly and informatively evaluate the running time performance of complex solving pipelines on problems from different application domains, and ultimately best employ these pipelines for solving real-world SCPs?

In addressing this question, we make the following additional contribution:

C7 We apply the paradigm of *programming by optimisation (PbO)* [80] to all solving pipelines described in this paper. For most of our design choices we implement alternatives and/or expose parameters to make the pipelines maximally configurable. We then use *automated algorithm configuration (AAC)* [79] to find optimised configurations of these solving pipelines. To the best of our knowledge, this work represents the first instance of using first use of AAC in exact probabilistic inference.

[1] Available at `www.gecode.org`.
[2] Available at `www.gurobi.com`.

The remainder of this work is organised as follows. Part I serves to provide some background for the reader. Specifically, we provide some background on how chance, logic and reasoning are related to each other in Chapter 2. We then describe several programming paradigms for optimisation in Chapter 3.

Part II is dedicated to the contributions of this work. Specifically, we start Part II with a detailed description of the SCPs that we study in this work, and briefly discuss related problems in Chapter 4. In that chapter, we also describe the problem settings used in our experiments, and the data on which those problems are formulated.

In Chapter 4, we also show how we use the *stochastic constraint on probability distributions (SCPD)* formulated in Section 1.2 to model these problems (**C1**) and present the new programming language that we propose specifically for modelling SCPs (**C2**). Then, in Chapter 5 we propose a specific kind of exact SCP solving method that takes a stochastic constraint on an SDD or OBDD encoding and *decomposes* it into a multitude of smaller constraints, resulting in a CP or MIP model that is then solved with an off-the-shelf CP or MIP solver (**C3** and **C4**). These parts of Chapter 4 and all of Chapter 5 are based on research previously published in:

Anna L.D. Latour, Behrouz Babaki, Anton Dries, Angelika Kimmig, Guy Van den Broeck, and Siegfried Nijssen. 'Combining Stochastic Constraint Optimization and Probabilistic Programming: From Knowledge Compilation to Constraint Solving'. In: *Principles and Practice of Constraint Programming: 23rd International Conference (CP 2017)*. 2017, pp. 495–511.

In the next chapter, Chapter 6, we note that the decomposition approach does not guarantee GAC and that a trivial modification of this approach does not significantly improve performance (**C5**). We therefore propose a new, global SCMD, which operates on OBDD encodings of probability distributions with a specific *monotonic* property, and demonstrate its superior performance (**C6**). Chapter 6 is based on research previously published in:

Anna Louise D. Latour, Behrouz Babaki, Siegfried Nijssen. 'Stochastic Constraint Propagation for Mining Probabilistic Networks'. In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI 2019)*. 2019, pp. 1137–1145.

We then take the decomposition method described in Chapter 5 and the global constraint propagation method described in Chapter 6, and apply the paradigm of PbO to these methods in Chapter 7. We implement alternative design choices

for the different elements of the SCP solving pipelines and use AAC to automatically configure them on sets of problems instances from several applications domains, demonstrating that the global SCMD propagation algorithm from Chapter 6 tends to outperform the other methods (**C7**). This chapter is based on research previously presented in:

- Daniël Fokkinga, Anna Louise D. Latour, Marie Anastacio, Siegfried Nijssen, and Holger Hoos. 'Programming a Stochastic Constraint Optimisation Algorithm, by Optimisation'. In: *Data Science meets Optimization workshop 2019 (DSO 2019), co-located with IJCAI 2019, Macao, 2019*.

- Anna L.D. Latour, Behrouz Babaki, Daniël Fokkinga, Marie Anastacio, Holger H. Hoos, and Siegfried Nijssen. 'Exact Stochastic Constraint Optimisation with Applications in Network Analysis'. In: *Artificial Intelligence, vol 304, 2022.*