Automated Reasoning For Uncertain Markov Processes

by

Muhammad Maaz

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy

Department of Mechanical and Industrial Engineering
University of Toronto

Automated Reasoning For Uncertain Markov Processes

Muhammad Maaz
Doctor of Philosophy

Department of Mechanical and Industrial Engineering
University of Toronto
2025

Abstract

Markov processes are a ubiquitous modeling tool used to model a variety of probabilistic phenomena across a range of disciplines. In this thesis, we develop algorithms for studying Markov processes using tools from the field of *automated reasoning*.

First, we tackle a variant of sensitivity analysis, where we want to obtain all possible parameters for a Markov process such that the total reward achieves a fixed threshold. The set of such parameters forms a semialgebraic set, and so we use cylindrical algebraic decomposition, an algorithm developed for the existential theory of the reals, to describe this set. By exploiting properties of our polynomial system, we develop a variant that runs in singly exponential time, instead of the doubly exponential complexity in the general case.

Next, we study properties of Markov processes where the parameters are functions, given as machine learning models, of exogenous variables. Using ideas from formal verification of machine learning models and probabilistic model checking, we show how to obtain guaranteed bounds on the behavior of such processes. For a wide selection of machine learning models, we show that obtaining such guarantees is equivalent to solving a bilinear program, which are classically NP-hard problems. We develop a special decomposition algorithm that solves the bilinear program orders-of-magnitude faster than state-of-the-art solvers.

Our algorithmic developments are implemented in two software packages: `markovag`, which implements our variant of cylindrical algebraic decomposition; and `markovml`, which provides a domain-specific language for constructing Markov processes, embedding pretrained machine learning models, and then solving the resulting bilinear program with our decomposition scheme.

Lastly, we perform a comprehensive cost-effectiveness analysis using data from nearly 25,000 real cardiac arrests in Ontario, Canada. Using drones to deliver defibrillators to the site of cardiac arrests has been trialled, but a cost-effectiveness analysis has been lacking. Our detailed analysis shows that drones are a cost-effective solution, and our findings are robust to modeling assumptions. The richness of this analysis makes it a fertile case study for our novel algorithms, and we show how our techniques provide deeper insights than usual cost-effectiveness methods.

To my mom and dad, who raised me to be curious.

Acknowledgements

First and foremost, I thank my supervisor, Timothy Chan. It is not an overstatement to say that Tim changed my life. Tim took a chance on me as a PhD student despite coming from a non-traditional background. He gave me the space to explore risky ideas, trusted me with intellectual freedom and wide latitude in research directions. He has always been extremely available to discuss research or other professional concerns. I am also indebted to him for funding my many conference travels during my PhD.

The ideas in this thesis would not have been possible without the support of my committee. Thank you to Xujie Si, whose automated reasoning class I took on a whim but ended up changing the course of my research career, leading to this very thesis. Thank you to Eldan Cohen, whose broad knowledge always led to incisive questions during our meetings. I would not have pursued a PhD without the support of my undergraduate research supervisor, Anastasios Papanastasiou, who first showed me the wonderful world of research. I am grateful to the Natural Sciences and Engineering Research Council of Canada for their generosity in awarding me a Vanier Scholarship, which allowed me to fully focus on doing good research.

Next, I thank my dear friends for their support during my PhD. Thanks especially to Aahil Dayani, with whom watching movies was always a welcome respite; Talha Tahir, who would always support my eccentric life plans without judgment; Roshan Naufal, who would entertain me despite being thousands of kilometers away; Baljit Sohi, who has an endless dedication to finding third spaces; Madhu Gunasingam, with whom I'd talk about math until 3am at our local shawarma spot; Niloy Anjum, who kept me grounded with our Friday night weightlifting sessions; Mathepan Mahendralingam, who was always available for any sort of advice; and Navid Khan, who was always willing to go on a late night drive. There are several dozen other people whose support and friendship over the last few years kept me sane, among them, in no order, Ali Butt, Christopher Yao, Amar Camlasaran, Fahaam Tashfeen, Melanie Wong, and Fuad Ali.

The Applied Optimization Lab was a great place for my intellectual growth these last few years. I would like to especially thank Bo Lin, who was a wonderful mentor to me; Jesse Ward-Bond, whose organizational skills I wish I could emulate; Vinicius Jameli, who was always an injection of positivity; as well as Rachel Wong, Craig Fernandes, and everyone else.

I spent 6 incredible months during my PhD at the Massachusetts Institute of Technology, and I thank Dimitris Bertsimas for hosting me. I'll always fondly remember my time there. Thanks to Jennifer Lin, whom I would have the most interesting conversations with for endless hours, and whom I'm counting on to become a billionaire; Salman Hasib, a truly unexpected and wise friend; and, my cousin Najaf Khan and her husband Ali Zaidi, who made Boston feel like home.

This thesis is, of course, dedicated to my parents, but I am also grateful to my brother, Moeez Muhammad, for our valuable intellectual discussions; my cousin Zeshan Khan, who pushed me to be more ambitious; and the rest of my extended family.

Lock yourself in a room
Doing five beats a day for three summers

— Kanye West

Contents

Chapter 1

Introduction

1.1 Motivation

Markov processes, which model probabilistic transitions between states, are fundamental mathematical models used across operations research, computer science, engineering, and healthcare [Stewart, 2021]. In computer science, they capture the behavior of probabilistic systems such as hardware, communication protocols, or autonomous agents [Baier and Katoen, 2008]. In engineering, they are used to analyze degradation and failure in complex machinery [Rausand and Hoyland, 2003]. In healthcare, they model patient transitions between clinical states and underpin cost-effectiveness analyses (CEA) of medical interventions [Sonnenberg and Beck, 1993].

This thesis is most particularly inspired by the use of Markov processes in health economics [Rudmik and Drummond, 2013]. A new medical intervention would be evaluated based on how it alters the primitives of the Markov reward process. For example, a promising intervention might lower the transition probability to worse health states or increase rewards associated with quality of life in healthier states. After calculating lifetime benefits and costs using the model, a policymaker may be interested in whether the total reward associated with the intervention exceeds a given threshold, or whether the reward of one intervention exceeds that of an alternative. Other metrics of interest in a CEA may include the net monetary benefit (benefit multiplied by willingness-to-pay per unit benefit minus cost) or the incremental cost-effectiveness ratio (difference in costs of two interventions divided by their difference in benefits).

The classic study of Markov processes (which we will sometimes, loosely, also refer to as Markov chains or Markov models) assumes that all parameters are known exactly. This setting has been deeply studied, and their properties can be found in standard textbooks, e.g., Puterman [2014] or Rosenthal [2019]. However, when they are used to model the real-world, several difficulties arise. Most real-world systems do not have known parameters. There may be significant variability that can be modeled in various ways: e.g., parameters may lie in intervals, or some more complicated set, be drawn from a probability distribution, or be given by a function of exogenous variables which themselves lie in some set. For example, in the healthcare context, the transition probabilities may be taken from empirical data, which has inherent uncertainty, or be a function of the patients' characteristics.

Under such uncertainty, we would ideally like to obtain some rigorous guarantees of the behavior of these Markov models. This can be analyzed through the lens of *automated reasoning*. Automated reasoning is a broad field of computer science that aims to provide mathematical proofs, derived by computers, about the behavior of a program or a system [Portoraro, 2025]. It is used in practice to ensure necessary properties of hardware and software alike, which is called *formal verification*. In recent times, formal verification has even been applied to machine learning models. Automated reasoning spans several different subdomains: it includes symbolic manipulation such as with computer algebra systems, interactive theorem provers like Lean [De Moura and Ullrich, 2021] (which have garnered increasing interest in recent years from being used in conjunction with large language models), and automated theorem provers like Z3 [De Moura and Bjørner, 2008]. Programs like Z3 are solvers for Boolean satisfiability (SAT) problems [Papadimitriou and Steiglitz, 2013], which can encode logical propositions, and extensions known as satisfiability modulo theories (SMT), where Boolean clauses are replaced by clauses under some other "theory", e.g., linear inequalities. To see how solving an SMT problem can be used to prove a theorem, consider the simple case of proving the following property of quadratics: a quadratic $ax^2 + bx + c$ with $a > 0$ and $b^2 - 4ac < 0$ (i.e., a convex quadratic with negative discriminant) has no non-positive values. This is equivalent to saying that $\forall a \forall b \forall c (a > 0 \wedge b^2 - 4ac < 0 \implies \forall x (ax^2 + bx + c > 0))$, which can be expressed in Z3, and Z3 will tell us that this theorem is indeed true. To do so, Z3 will use a combination of SAT solving algorithms like conflict-driven clause learning, and a theory solver which can reason about the nonlinear inequalities that make up the clauses of the expression.

Hence, underlying automated theorem provers are solvers that can work with clauses like linear or polynomial inequalities. There is thus a tight connection between optimization and automated reasoning. For example, Z3 would use a linear programming algorithm like simplex if it was given a satisfiability problem containing linear inequalities. As well, in order to verify a property of some system, we often want to bound the "worst-case", which can be done by solving some optimization problem. Driven by the desire to solve satisfiability problems, the computer science community has pushed the boundaries of what sorts of problems can be decided. While satisfiability of linear inequalities has been known to be solvable for quite some time – first with Gaussian elimination for equalities, which in fact goes back to Newton and possibly before [Grcar, 2011], then with Fourier-Motzkin elimination for inequalities [Fourier, 1827], and then with the development of linear programming in the 20th century by Dantzig (see, e.g., Dantzig et al. [1955]), von Neumann, Kantorovich, and others – the case for *polynomial* inequalities is much harder. It was not even known to be decidable until Tarski [1951], and the first algorithm only came out in the 1970s with Collins [1976], both of whom were motivated by solving the problem of the "existential theory of the reals", i.e., to determine if a set defined by polynomial inequalities is empty.

This is important because, as we will show throughout this thesis, studying properties of Markov processes with uncertain parameters can be done by studying an appropriate nonlinear system. Automated reasoning thus provides us with powerful tools to understand them. There is a direct link between the tools developed in that field and what we will do in this thesis. In particular, in Chapter 2, we will apply the algorithm of cylindrical algebraic decomposition [Collins, 1976], which is what Z3 and similar programs use to reason about polynomial inequalities, to perform an exact sensitivity analysis of Markov models. In Chapter 3, we will embed machine learning models into our Markov models, and show how to obtain bounds on the behavior of this system by solving a nonlinear optimization problem.

Concretely, the problems that we study in this thesis are both problems seen in health economics. Namely, Chapter 2 tackles the problem of *sensitivity analysis*, i.e., seeing if the results of a health economic analysis are sensitive to perturbations in the parameters, and Chapter 3 can be understood to be tackling the problem of *subgroup analysis*, i.e., analyzing outcomes for a group of patients defined by their features, such as women over the age of 65. As will be discussed further in the related work, currently, both of these problems are handled in a rather non-rigorous way in the health economics community, namely relying on sampling and simulations.

Both of these problems are directly inspired by the work in Chapter 4. In that chapter, we show how Markov models can be used to model a complicated real-world medical problem: simulating the effect of drone-delivered medical supplies on patient outcomes. To be more specific, when a cardiac arrest happens at home, defibrillation should be given as soon as possible. Hence, some groups around the world have begun to study using drones to fly defibrillators to the site of a cardiac arrest. In this thesis, we present the first cost-effectiveness study of drones for defibrillator delivery, using data on nearly 25,000 real patients in Ontario. This analysis comprises multiple components, including facility location optimization, machine learning, and Markov models. It thus inspired the work in Chapters 2 and 3. As little is known about real-world drone operation, there is significant uncertainty in some of the inputs of this analysis, which directly inspired the methods in Chapter 2 for doing a rigorous sensitivity analysis. Because machine learning models are used as inputs into the Markov models, this inspired the algorithm in Chapter 3 for rigorously analyzing the properties of such systems.

1.2 Summary of Contributions

To summarize the contributions in this thesis:

1. In Chapter 2, we develop a technique for exact sensitivity analysis of Markov reward processes using cylindrical algebraic decomposition (CAD). We study how to analyze the set of parameters such that the total reward satisfies an inequality. Our main technical contribution is the development of a variant of CAD, that, under the special structure arising from this class of polynomial systems, is drastically more efficient than a general CAD. We also implement our algorithm in a software package, `markovag`, which is based on SymPy as the underlying computer algebra system.

2. In Chapter 3, we study how to formally verify properties of Markov processes such as bounding the total reward, reachability, or hitting time, when the parameters are given by the outputs of machine learning (ML) models. This is important as, e.g., we may want individualized parameters that are determined by a pretrained ML model. We show that for a broad class of ML models, this problem is equivalent to a bilinear program, and we develop a novel decomposition algorithm to solve it, which we show is up to 100x faster than current bilinear solvers. We develop a powerful software package, `markovml`, which provides a domain-specific language for building Markov processes and integrating ML models, and then verifying their properties.

3. In Chapter 4, we perform the first cost-effectiveness analysis of drones for delivering automated external defibrillators. Using data on nearly 25,000 real cardiac arrests, we fit nearly one

thousand optimal drone networks, across a variety of objectives and network sizes. We then analyze each drone network by simulating patient outcomes as if the drone network had been present, and use a Markov model to track patients over time. We find that drones are a cost-effective intervention, and this holds true across extensive robustness checks.

1.3 Thesis Organization

This thesis compiles work that has been published across three different papers. Chapter 2 is the paper Chan and Maaz [2024], Chapter 3 is the paper Maaz and Chan [2025], and Chapter 4 is the paper Maaz et al. [2025]. The first two papers share similar mathematical setup as well as some related literature, so we provide these in the end of this chapter. Both Chan and Maaz [2024] and Maaz and Chan [2025] contain case studies that re-analyze Maaz et al. [2025], but in this thesis, we separate them and relegate these re-analyses to Chapter 4. There are some slight differences from the published individual papers[1].

1.4 Related Work

In this section, we go over the related work that is common to both Chapters 2 and 3. Both are intimately connected to the literature on uncertain Markov processes as well as cost-effectiveness analyses in healthcare.

1.4.1 Uncertainty in Markov Processes

Markov processes with parameters in uncertainty sets or defined by differentiable functions have been well-studied [Caswell, 2019, Dai, 1996, Caswell, 2013, Hermans and De Cooman, 2012, Blanc and den Hertog, 2008, De Cooman et al., 2014, Marbach and Tsitsiklis, 2001, 2003]. For example, De Cooman et al. [2014] develop a generalization of the Perron-Frobenius Theorem when the transition probabilities lie within a credal set. Blanc and den Hertog [2008] study Markov chains with row-wise uncertainty to compute bounds on hitting times and stationary distributions. However, this line of research focuses on Markov chains, not Markov reward processes, and as a result is silent on derived metrics like the total reward. There is also a large literature on Markov *decision* processes with uncertain parameters [Nilim and El Ghaoui, 2005, Iyengar, 2005, Delage and Mannor, 2010, Wiesemann et al., 2013, Goyal and Grand-Clément, 2023, Grand-Clément and Petrik, 2024], but in this thesis we are not studying Markov decision processes, i.e., there is no policy.

[1]Proofs are always provided directly after the statement of the result. Additional details, e.g., about experiments, which are in the appendices of the original papers, are instead integrated into the chapter, as are expository material; although the several supplementary figures and tables for Chapter 4 are in the appendix. Some of the introductions are slightly modified, due to the combined introduction written in this chapter, and some text in this chapter is taken verbatim from the introductions of Chan and Maaz [2024] and Maaz and Chan [2025]. Some section titles may differ in order to give a unified style across the different chapters. There may also be slight modifications to transition sentences to reflect the flow of ideas. This thesis is meant to be a self-contained body of work, and my goal is to make it instructive to future readers. Hence, Chapter 2 contains a full exposition of how cylindrical algebraic decomposition works from the very basics, which is difficult to find in the literature; and Chapter 3 contains a full tutorial of the software package, `markovml`, developed in Maaz and Chan [2025]. Chapter 4, being based on a paper published in a medical venue, is written rather differently than Chapters 2 and 3, i.e., there is very little math, except for towards the end of the chapter when we demonstrate the re-analyses using our new algorithms.

The paper that most directly inspired the work in this thesis is that of Goh et al. [2018], who develop a method for finding the maximum or minimum infinite horizon total reward for a Markov reward process, where the set of probability matrices has a row-wise structure. They show that solving such a problem can be done by formulating a certain Markov decision process, and then solving it using standard policy iteration. Our work builds on this in the following two ways. First, simply knowing the extrema would be insufficient if the threshold lies between the maximum and minimum, as the policymaker would not know over which parameter values the threshold is met. In contrast, our work in Chapter 2 allows a policymaker to know exactly which parameter values attain or violate a given inequality. Second, when we integrate ML-driven parameters as in Chapter 3, we cannot rely on the special row-wise structure, and hence have to develop a more general technique, which in this thesis is to leverage bilinear programming.

1.4.2 Cost-Effectiveness Analysis

Markov models have been a ubiquitous tool for cost-effectiveness analysis in healthcare [Carta and Conversano, 2020]. They are part of the larger class of multistate models [Hougaard, 1999] that model transitions between health states. These models compute metrics like the incremental cost-effectiveness ratio (ICER) and net monetary benefit (NMB) [Sonnenberg and Beck, 1993]. Markov models in healthcare are typically analyzed either using linear algebra or via cohort simulations [Sonnenberg and Beck, 1993]. As these models exhibit parameter uncertainty, sensitivity analysis is an essential step [Briggs et al., 1994, Jain et al., 2011, Rudmik and Drummond, 2013]. Indeed, it is recommended by health economics professional societies [Briggs et al., 2012] and even mandated by policymakers [Andronis et al., 2009]. However, when sensitivity analyses are limited to one or two parameters, larger interaction effects or correlations may be missed [Vreman et al., 2021]. Furthermore, the range of parameter values to test are often chosen arbitrarily. In contrast, the method we present in Chapter 2 can perform arbitrary multi-way sensitivity analyses, uncovering interactions between parameters, and will identify the full range of parameter values that lead to a cost-effectiveness result holding.

As well, we often want to analyze subgroups of patients, or to identify heterogeneity between patients. In the healthcare literature, these are typically studied using Markov "microsimulation" models, which capture heterogeneous patient trajectories over time [Krijkamp et al., 2018], using individual-level transition probabilities derived from so-called risk scores – typically logistic regression, although other models are sometimes used [Mertens et al., 2022, Wilde et al., 2019, Lee et al., 2020, Breeze et al., 2017]. While existing approaches rely on Monte Carlo simulation, in Chapter 3 we instead provide an exact, non-simulation-based framework that supports a broad class of machine learning models, aligning with the shift toward more sophisticated analytics in healthcare.

1.5 Mathematical Setup

Notation Vectors are lowercase bold, e.g., $\mathbf{x}$, with i-th entry x_i, and matrices by uppercase bold letters, e.g., $\mathbf{M}$ with (i, j)-th entry M_{ij}. The identity matrix is denoted by $\mathbf{I}$, the vector of all ones by $\mathbf{1}$, with dimensions inferred from context. The set of integers from 1 to n is denoted by $[n]$.

A (discrete-time, finite-state) *Markov chain* with n states, indexed by $[n]$, is defined by a row-stochastic transition matrix $\mathbf{P} \in \mathbb{R}^{n\times n}$, where P_{ij} is the probability of transitioning from state i to state j, and a stochastic initial distribution vector $\boldsymbol{\pi} \in \mathbb{R}^n$, where π_i is the probability of starting in state i. Furthermore, if we assign rewards to each state, we call this a *Markov reward process*. A Markov reward process has a reward vector $\mathbf{r} \in \mathbb{R}^n$, where r_i is the reward for being in state i for one period. A state is *absorbing* if it cannot transition to any other state, and *transient* otherwise.

Below, we recount three of the key properties commonly computed in practice [Puterman, 2014].

Definition 1.1 (Reachability)**.** The probability of eventually reaching a set of states $S \subseteq [n]$, from the set $T \subseteq [n]$ of transient states, where $S \cap T = \emptyset$, is given by $\tilde{\boldsymbol{\pi}}^\top(\mathbf{I} - \mathbf{Q})^{-1}\mathbf{R}\mathbf{1}$, where $\mathbf{Q}$ is the transition matrix restricted to T, $\mathbf{R}$ is the transition matrix from T to S, and $\tilde{\boldsymbol{\pi}}$ is the initial distribution over T.

Definition 1.2 (Expected hitting time)**.** The expected number of steps to eventually reach a set of states $S \subseteq [n]$, from the set $T \subseteq [n]$ of transient states, where $S \cap T = \emptyset$, assuming that the chain will reach S from T with probability 1, is given by $\tilde{\boldsymbol{\pi}}^\top(\mathbf{I} - \mathbf{Q})^{-1}\mathbf{1}$, where $\mathbf{Q}$ is the transition matrix restricted to T, and $\tilde{\boldsymbol{\pi}}$ is the initial distribution over T.

Definition 1.3 (Total finite-horizon discounted reward)**.** The total finite-horizon discounted reward, over a finite horizon t, with a discount factor $\lambda \in (0, 1)$ is given by $\sum_{m=0}^{t} \boldsymbol{\pi}^\top \lambda^m \mathbf{P}^m \mathbf{r}$.

Definition 1.4 (Total infinite-horizon discounted reward)**.** The total infinite-horizon discounted reward, with a discount factor $\lambda \in (0, 1)$ is given by $\sum_{m=0}^{\infty} \lambda^m \boldsymbol{\pi}^\top \mathbf{P}^m \mathbf{r} = \boldsymbol{\pi}^\top \left(\mathbf{I} - \lambda \mathbf{P}\right)^{-1} \mathbf{r}$.

For each of the above quantities, we can restrict the analysis to a single state, e.g., for reachability starting from a state i, we simply set $\tilde{\pi}_i = 1$ and for $j \neq i$, set $\tilde{\pi}_j = 0$.

Chapter 2

Exact Sensitivity Analysis of Markov Reward Processes

2.1 Introduction

This chapter develops a new approach to conduct exact, deterministic sensitivity analyses of a Markov reward process. Our approach leverages ideas from algebraic geometry, particularly cylindrical algebraic decomposition (CAD), and applies them to the mathematical structure of common cost-effectiveness analyses, which enables exact analysis more efficiently than for general polynomial systems. The implication is that exact multi-way (where multiple parameters are varied simultaneously) sensitivity analysis is possible, which allows a policymaker to fully describe complex parameter regimes where new technologies or medical interventions are cost-effective.

As the parameters of a Markov process may be subject to significant uncertainty, it is common to do a sensitivity analysis, e.g., in a cost-effectiveness analysis (CEA). A typical approach is a one-way deterministic sensitivity analysis, which means that one parameter is varied within a range or set of values while all others are fixed at some nominal value. This approach is straightforward but does not capture the joint effect of multiple parameters. Multi-way sensitivity analyses create a multi-dimensional grid of test points over the parameter space, which measures the impact of parameter interactions, but quickly becomes intractable in the number of parameters tested and grid granularity. In a systematic review of CEAs, Jain et al. [2011] found that 86% of studies in their sample conducted a one-way sensitivity analysis, but only 45% conducted a multi-way sensitivity analysis, likely owing to these difficulties. In practice, multi-way sensitivity analyses rarely extend beyond two parameters [Briggs et al., 1994]. We note that there is another type of sensitivity analysis known as probabilistic sensitivity analysis, where parameter values are drawn from distributions and their joint effect simulated [Baio and Dawid, 2015]. However, they face a similar issue as parameter distributions are often arbitrarily chosen. In contrast, our method makes no assumptions on the ranges nor distributions of the parameters, and instead enumerates the full range of parameters that yields the desired result.

We make the key observation that the questions typically asked in a cost-effectiveness analysis based on a Markov reward process can be described as a system of polynomial inequalities. Determining whether an intervention remains cost-effective if parameter values vary within given intervals is thus equivalent to determining whether a polynomial system satisfies a set of inequalities over those parameter intervals. Hence, we study deterministic sensitivity analysis through the lens of algebraic geometry, which provides tools that facilitate analysis of multivariate polynomial systems, such as *cylindrical algebraic decomposition* [Collins, 1974]. CAD was the first practical algorithm for solving systems of polynomial inequalities and works by decomposing the multidimensional real space into cylindrical cells over which each polynomial is sign invariant. Once these cells are defined, the algorithm can easily check feasibility of the polynomial system over each cell [Basu et al., 2006]. In doing so, we obtain a tree-like representation of the whole space over which the polynomial system holds. In this chapter, we will use CAD to fully represent the cost-effectiveness region in a multi-way sensitivity analysis. While analyzing general polynomial systems using CAD remains computationally challenging, the polynomial systems of interest in a Markov reward process-based CEA can be analyzed much more tractably.

Our main contributions are as follows.

1. *Semialgebraic representation of Markov reward processes.* We show that sensitivity analysis of common cost-effectiveness analysis quantities, including bounding or comparing total benefits, total costs, incremental cost-effectiveness ratios, and net monetary benefits, is equivalent to determining whether certain polynomial systems are feasible. Thus, describing the cost-effective parameter space can be done by analyzing the CAD of these systems.

2. *Cylindrical algebraic decomposition.* We demonstrate that the polynomial systems induced by the aforementioned analyses belong to a special class that makes the CAD construction more efficient. This class accommodates important considerations such as the transition probability matrix possessing the increasing failure rate property. We develop a specialized version of the general CAD algorithm for this class of systems. We show that this CAD has a singly exponential size, compared to the doubly exponential size in a general CAD.

3. *Software.* We develop a Python package, `markovag`, that implements our algorithms. It can construct polynomial systems representing sensitivity analysis of common CEA metrics, analytically characterize the boundary in a multi-way sensitivity analysis, and construct the CAD.

4. *Case studies.* To demonstrate the CAD approach to sensitivity analysis, we apply our algorithms and software in two case studies. The first case study uses synthetic data to show that a traditional parameter grid search could easily mischaracterize a non-linear cost-effectiveness boundary. The second study, in Chapter 4, we re-analyze a real CEA from the literature and show that our approach reveals a larger cost-effective parameter space than in the original analysis and elucidates relationships between model parameters that would not be otherwise obvious.

2.2 Related Work

This chapter relates to three different bodies of literature, across the domains of health economics, stochastic processes, and algebraic geometry. The first two are discussed earlier in Section 1.4. The third one is covered here.

2.2.1 Cylindrical Algebraic Decomposition

A cylindrical algebraic decomposition (CAD) is a tree-like decomposition of the real space representing the solutions to a polynomial system, which is how we apply it in this chapter. The algorithm to construct a CAD tree was initially developed by Collins [1974] to solve the problem of the existential theory of the reals, which asks for a satisfying assignment of real numbers to a Boolean combination of polynomials. This problem was known to be solvable theoretically by Tarski [1951] but it was not until Collins [1974] that a practical algorithm was developed. This problem is important in computer science as it fits into the general framework of satisfiability modulo theories (SMT), a generalization of the classical Boolean satisfiability problem (SAT) [Papadimitriou and Steiglitz, 2013] to statements with variables that can be numbers or even data structures [De Moura and Bjørner, 2011]. Solving polynomial systems with CAD has long been used in robotics motion planning [Canny, 1987, Schwartz and Sharir, 1990].

More broadly, there are some strong connections between techniques from algebraic geometry and the operations research literature, particularly for polynomial optimization and semidefinite programming [Lasserre, 2001, Parrilo, 2003, Blekherman et al., 2012, Parrilo and Thomas, 2019]. However, to date, techniques from this field have not been applied to the study of Markov reward processes. Our technical contribution will be to apply concepts from algebraic geometry to study the solutions to our class of polynomial systems, derived from Markov reward process sensitivity analysis.

The computer algebra literature has identified special cases of polynomial systems for which CAD can be simplified. There is a recent body of work on speeding up CAD when the system contains equalities [England et al., 2020], which do arise in the class of systems we study, but the algorithm remains asymptotically doubly exponential. Incremental CAD [Kremer and Ábrahám, 2020] extends a CAD of a polynomial system to a CAD of the same system with an additional inequality. We use a similar idea of extending a CAD by incrementally adding inequalities in our algorithm. Lastly, Strzeboński [2010] developed an algorithm to construct a CAD from Boolean formulas of CADs. Our contribution to computer algebra is the identification of a special but broad class of polynomial systems under which we can construct the CAD more tractably.

2.3 Semialgebraic Representations of Markov Reward Processes

We consider, as in Section 1.5, a Markov reward process with n states, transition probability matrix $\mathbf{P} \in \mathbb{R}^{n\times n}$, reward vector $\mathbf{r} \in \mathbb{R}^n$, initial state distribution $\boldsymbol{\pi} \in \mathbb{R}^n$, and a discount rate $\lambda \in (0, 1)$.

We will study inequalities involving the finite and infinite-horizon rewards. As they will figure heavily in this chapter, we introduce the following notation to refer to them, taking their definitions from Section 1.5. The expected discounted reward over a finite horizon of length t, R_t, or over an infinite horizon, R_∞, are:

$$R_t := \sum_{m=0}^{t} \boldsymbol{\pi}^\top \lambda^m \mathbf{P}^m \mathbf{r} \tag{2.1}$$

and

$$R_\infty := \boldsymbol{\pi}^\top (\mathbf{I} - \lambda \mathbf{P})^{-1} \mathbf{r}. \tag{2.2}$$

For the purposes of sensitivity analysis, we consider R_t and R_∞ as functions of $\boldsymbol{\pi}$, $\mathbf{P}$, and $\mathbf{r}$, since these are the quantities most likely to be estimated from data and subject to uncertainty. We assume that λ is fixed. Clearly, R_t is a polynomial in $\boldsymbol{\pi}$, $\mathbf{P}$, and $\mathbf{r}$. For R_∞, we can re-write the matrix inversion as

$$R_\infty = \frac{1}{\det(\mathbf{I} - \lambda\mathbf{P})} \boldsymbol{\pi}^\top \operatorname{adj}(\mathbf{I} - \lambda\mathbf{P})\mathbf{r}, \tag{2.3}$$

where $\det(\cdot)$ is the determinant and $\operatorname{adj}(\cdot)$ is the adjugate matrix operator [Strang, 2022]. In this form, there are three key properties of R_∞ that we will use.

Lemma 2.1. *R_∞ is a ratio of two polynomials in $\boldsymbol{\pi}$, $\mathbf{P}$, and $\mathbf{r}$. Furthermore, $\det(\mathbf{I} - \lambda\mathbf{P}) > 0$ and the adjugate $\operatorname{adj}(\mathbf{I} - \lambda\mathbf{P})$ has all non-negative entries.*

Proof. Let $\mathbf{M} = \mathbf{I} - \lambda\mathbf{P}$. Each element of the adjugate matrix is the determinant of a submatrix of $\mathbf{M}$, and the determinant is a multilinear map, and so each element of $\operatorname{adj}\mathbf{M}$ is a polynomial of a submatrix of $\mathbf{P}$. Thus $\boldsymbol{\pi}^\top \operatorname{adj}(\mathbf{M})\mathbf{r}$ is a polynomial of $\boldsymbol{\pi}$, $\mathbf{P}$, and $\mathbf{r}$. As noted, $\det\mathbf{M}$ is a polynomial of $\mathbf{P}$, so R_∞ is a ratio of two polynomials. Next, $\mathbf{M}$ has the property that all of its real eigenvalues are positive [Berman and Plemmons, 1994, Theorem 2.3 in Chapter 6]. Since any complex eigenvalues come in conjugate pairs, the product of all eigenvalues, which equals the determinant, is positive. Lastly, the inverse of $\mathbf{M}$ has all non-negative entries [Berman and Plemmons, 1994, Theorem 2.3 in Chapter 6], and we already know its determinant is positive, so the adjugate of $\mathbf{M}$ has all non-negative entries. $\square$

Example 2.2. *In a Markov chain with $n = 2$ states, $\det(\mathbf{I} - \lambda\mathbf{P}) = 1 + \lambda^2 p_{11}p_{22} - \lambda^2 p_{12}p_{21} - \lambda p_{11} - \lambda p_{22}$ and $\operatorname{adj}(\mathbf{I} - \lambda\mathbf{P})$ is*

$$\begin{bmatrix} 1 - \lambda p_{22} & \lambda p_{12} \\ \lambda p_{21} & 1 - \lambda p_{11} \end{bmatrix}.$$

Hence, R_∞ is the following ratio of polynomials in $\mathbf{P}$, $\boldsymbol{\pi}$, and $\mathbf{r}$:

$$\frac{r_1(\lambda p_{21}\pi_2 + \pi_1(1 - \lambda p_{22})) + r_2(\lambda p_{12}\pi_1 + \pi_2(1 - \lambda p_{11}))}{1 + \lambda^2 p_{11}p_{22} - \lambda^2 p_{12}p_{21} - \lambda p_{11} - \lambda p_{22}}.$$

Sensitivity analyses of Markov reward processes, and cost-effectiveness analyses in particular, are typically concerned with identifying a range of input values over which inequalities involving R_t and R_∞ hold. Given the forms of R_t and R_∞ provided above, particularly in Lemma 2.1 for R_∞, typical inequalities associated with sensitivity analysis can all be written as polynomial inequalities. In the examples that follow, we focus on R_∞; the application to R_t is straightforward. Importantly, since the denominator of R_∞ is always positive, the sign of the inequality remains unchanged as elements of $\mathbf{P}$ are varied.

Total reward. Given a threshold $\gamma \in \mathbb{R}$, a policymaker may be interested in

$$\boldsymbol{\pi}^\top (\mathbf{I} - \lambda\mathbf{P})^{-1}\mathbf{r} \geq \gamma, \tag{2.4}$$

which can be written as the polynomial inequality

$$\boldsymbol{\pi}^\top \operatorname{adj}(\mathbf{I} - \lambda\mathbf{P})\mathbf{r} - \gamma \det(\mathbf{I} - \lambda\mathbf{P}) \geq 0. \tag{2.5}$$

This approach can be extended to the comparison of two interventions, labeled a and b. Determining whether the total reward associated with intervention a is greater than that of b

$$\boldsymbol{\pi}_a^\top(\mathbf{I}-\lambda\mathbf{P}_a)^{-1}\mathbf{r}_a \geq \boldsymbol{\pi}_b^\top(\mathbf{I}-\lambda\mathbf{P}_b)^{-1}\mathbf{r}_b \tag{2.6}$$

can be written as the polynomial inequality

$$\boldsymbol{\pi}_a^\top \operatorname{adj}(\mathbf{I}-\lambda\mathbf{P}_a)\mathbf{r}_a \det(\mathbf{I}-\lambda\mathbf{P}_b) - \boldsymbol{\pi}_b^\top \operatorname{adj}(\mathbf{I}-\lambda\mathbf{P}_b)\mathbf{r}_b \det(\mathbf{I}-\lambda\mathbf{P}_a) \geq 0. \tag{2.7}$$

Comparing more than two interventions would result in a system of polynomial inequalities.

Net monetary benefit (NMB). The net monetary benefit is defined as a constant willingness-to-pay threshold (W) for one unit of benefit, multiplied by the infinite horizon benefit, and then subtracting the infinite horizon cost. Let $\mathbf{b}$ and $\mathbf{c}$ be the vectors representing the one-period benefit and cost for each state. Then NMB equals

$$W\boldsymbol{\pi}^\top(\mathbf{I}-\lambda\mathbf{P})^{-1}\mathbf{b} - \boldsymbol{\pi}^\top(\mathbf{I}-\lambda\mathbf{P})^{-1}\mathbf{c} = \boldsymbol{\pi}^\top(\mathbf{I}-\lambda\mathbf{P})^{-1}(W\mathbf{b}-\mathbf{c}). \tag{2.8}$$

The right-hand side expression is equivalent to the total reward when $\mathbf{r} = W\mathbf{b} - \mathbf{c}$. Hence, an inequality that bounds the NMB can be written similarly to (2.5).

Incremental cost-effectiveness ratio (ICER). The incremental cost-effectiveness ratio is defined as the ratio between the difference in the infinite horizon costs and the difference in the infinite horizon benefits of two interventions. For two interventions a and b, the ICER is

$$\frac{\boldsymbol{\pi}_a^\top(\mathbf{I}-\lambda\mathbf{P}_a)^{-1}\mathbf{c}_a - \boldsymbol{\pi}_b^\top(\mathbf{I}-\lambda\mathbf{P}_b)^{-1}\mathbf{c}_b}{\boldsymbol{\pi}_a^\top(\mathbf{I}-\lambda\mathbf{P}_a)^{-1}\mathbf{b}_a - \boldsymbol{\pi}_b^\top(\mathbf{I}-\lambda\mathbf{P}_b)^{-1}\mathbf{b}_b}. \tag{2.9}$$

Bounding the ICER by γ results in a polynomial inequality after rearranging terms.

Remark 2.3 (Death state). Many Markov chains used in health economic models include an absorbing "death" state: once the process transitions there it remain in this state with probability 1 and reward 0. In this case, the infinite horizon expected total reward *without* discounting is finite and can be written as [Puterman, 2014]:

$$\bar{\boldsymbol{\pi}}^\top(\mathbf{I}-\mathbf{Q})^{-1}\bar{\mathbf{r}}, \tag{2.10}$$

where $\mathbf{Q}$ is a $(n-1)\times(n-1)$ matrix representing transitions between the transient states, and $\bar{\mathbf{r}}$ and $\bar{\boldsymbol{\pi}}$ are the subvectors corresponding to the rewards and initial distribution, respectively, on the transient states. Bounding this quantity can be similarly reformulated into a polynomial inequality. Note that the constraints on $\mathbf{Q}$ will be *substochastic*, i.e., the row sums need only be less than or equal to 1. As well, the row sums of $\mathbf{Q}$ need to be strictly greater than zero to ensure that $\mathbf{I}-\mathbf{Q}$ is invertible. Multiple absorbing states can also be accommodated easily. The takeaway is that our subsequent development for discounted Markov reward processes applies to this case as well.

2.4 Cylindrical Algebraic Decomposition

Motivated by the question *"for what values of $\boldsymbol{\pi}$, $\mathbf{P}$, and $\mathbf{r}$ does a given inequality hold?"*, this section studies general systems of polynomial inequalities. The focus is on those systems that possess relevant characteristics associated with Markov reward process analysis, as described in the previous section. For solving polynomial inequalities (i.e., identifying regions or points in the domain that satisfy the inequalities), the standard representation is a *cylindrical algebraic decomposition* (CAD). We provide an overview of CAD following Basu et al. [2006] and refer the reader to that book for more details.

A CAD of a polynomial system is a finite decomposition of $\mathbb{R}^k$ into disjoint *cells* where in each cell each polynomial in the system is sign-invariant[1]. Having such a decomposition allows us to easily test consistency of the system at a given point in the domain and describe the regions over which the system is consistent. Next, we formally define a cell.

Definition 2.4 (Cell)**.** A cell is defined recursively.

1. In $\mathbb{R}^1$, a cell is either an open interval or a point.
2. Let $k \geq 1$ and C be a cell in $\mathbb{R}^k$. In $\mathbb{R}^{k+1}$, a cell is either of the form $\{(x, y) \in \mathbb{R}^{k+1} \mid x \in C, f(x) < y < g(x)\}$ or $\{(x, y) \in \mathbb{R}^{k+1} \mid x \in C, y = f(x)\}$, where f and g are either algebraic functions[2] or $\pm\infty$, with $f(x) < g(x)$ for all $x \in C$.

Let $\mathbb{R}[\{x_i\}_{i=1}^k]$ be the ring of polynomials in $x_1, \ldots, x_k$. If we have a finite set of polynomials $F \subset \mathbb{R}[\{x_i\}_{i=1}^k]$, we call a CAD *adapted to* F if each $f \in F$ is sign-invariant (0, $+$, or $-$) over each cell. An important theorem from algebraic geometry is that for every finite set of polynomials F, there exists a CAD adapted to F (see Lojasiewicz [1965] or Basu et al. [2006, Theorem 5.6]).

Next, we consider inequalities involving elements of F. Let $f \in \mathbb{R}[\{x_i\}_{i=1}^k]$. An *$f$-atom* is an expression $f \bowtie 0$, where $\bowtie \in \{=, >, <, \geq, \leq\}$. A *semialgebraic set* is defined by Boolean combinations of f-atoms and is closed under union, intersection and complement. A corollary of the above theorem is that any semialgebraic set of f-atoms can be represented equivalently by a subset of the cells of the CAD adapted to F. Then, to test the consistency of a semialgebraic set, one can simply construct the CAD adapted to F and use a single point in each cell to test the system's consistency. Lastly, we can obtain an algebraic description of each cell to represent all the solutions of the system.

The recursive nature of the cells of a CAD make it intuitive to represent as a tree.

Example 2.5. *Consider the unit sphere $x^2 + y^2 + z^2 - 1 = 0$. Its simplest CAD is:.*

$$\begin{tikzcd}
x=-1 \arrow[d, dash] & & & -1<x<1 \arrow[dll, dash] \arrow[d, dash] \arrow[drr, dash] & & & x=1 \arrow[d, dash] \\
y=0 \arrow[d, dash] & y=-\sqrt{1-x^2} \arrow[d, dash] & & -\sqrt{1-x^2}<y<\sqrt{1-x^2} \arrow[dl, dash] \arrow[dr, dash] & & y=\sqrt{1-x^2} \arrow[d, dash] & y=0 \arrow[d, dash] \\
z=0 & z=0 & z=-\sqrt{1-x^2-y^2} & & z=\sqrt{1-x^2-y^2} & z=0 & z=0
\end{tikzcd}$$

The CAD can be constructed using straightforward geometric reasoning. We start with one-dimensional cells, so we project the sphere to the unit disc $x^2 + y^2 \leq 1$ in $\mathbb{R}^2$, and then again to the

[1]Different CADs can represent the same system, e.g., by splitting cells unnecessarily.

[2]An algebraic function is defined as being the zero of some polynomial. It is a strictly larger class of functions than the polynomials. For example, $\sqrt{x}$ is not a polynomial, but it is algebraic, as it is a solution to $y^2 - x = 0$. On the other hand, $\sin x$ is not algebraic.

interval $-1 \leq x \leq 1$ *in* $\mathbb{R}^1$*. Form cells* -1*,* $(-1, 1)$*, and 1. Given the* x *values in each cell, determine the feasible* y *values based on the unit disc. Repeat for the* z *values based on the unit sphere.*

A cell in $\mathbb{R}^3$ *is the conjunction of all the nodes in a path from* x *to* z*. An example is* $x = -1 \wedge y = 0 \wedge z = 0$*. By taking the disjunction of all cells, we get the full decomposition of the unit sphere.*

This simple example illustrates the general principles for how to construct a CAD for any polynomial system in $\mathbb{R}^k$: 1) Repeatedly project the polynomials to $\mathbb{R}^1$; 2) Create cells by identifying appropriate points (i.e., roots) and intervals; 3) Over each cell in $\mathbb{R}^1$, construct a cylinder in $\mathbb{R}^2$, and create cells in $\mathbb{R}^2$ by computing the roots and intervals of each projection in $\mathbb{R}^2$. 4) Repeat until we reach $\mathbb{R}^k$.

While the above approach is intuitive, it required the development of special projection operators [Collins, 1974, 1976] to allow systematic CAD construction for arbitrary systems. Next, we describe the algorithm for generating a CAD of a system. As CAD crucially hinges on projection operators with certain properties, we now describe the *Hong projection operator* in detail. Later, we will analyze the behavior of the Hong projection operator for our technical results.

2.4.1 Background on Hong Projection Operator

To describe Hong's projection operator [Hong, 1990], we need to introduce some new notation. To use a projection operator, we have to choose a main variable, or mvar. The polynomials will be treated as univariate polynomials in the mvar. Then, for a polynomial f, the leading term, $\mathrm{ldt}(f)$, is the term with the highest exponent of the mvar, the leading coefficient, $\mathrm{ldcf}(f)$, is the coefficient of $\mathrm{ldt}(f)$, and the degree, $\deg(f)$ is the highest exponent with which the mvar appears.

Two crucial concepts in algebraic geometry are the *reducta* and the *principal subresultant coefficients*, which we define below.

Definition 2.6 (Reducta)**.** The reductum of a polynomial f with a chosen mvar is $\mathrm{red}(f) = f - \mathrm{ldt}(f)$. We inductively define the ith level reductum as: $\mathrm{red}^0(f) = f$ and $\mathrm{red}^i(f) = \mathrm{red}(\mathrm{red}^{i-1}(f))$. Then, the reducta set $\mathrm{RED}(f) = \{\mathrm{red}^i(f), 0 \leq i \leq \deg(f), \mathrm{red}^i(f) \neq 0\}$.

Example 2.7. *Let* $f = x_1x_2 + x_3x_4 - 1$*, with mvar* x_1*. Then,* $\mathrm{red}^0 = x_1x_2 + x_3x_4 - 1$*, and* $\mathrm{red}^1 = x_3x_4 - 1$*. So,* $\mathrm{RED}(f) = \{x_1x_2 + x_3x_4 - 1, x_3x_4 - 1\}$*.*

Definition 2.8 (Principal subresultant coefficients)**.** Let there be two polynomials f, g, with degrees p, q, respectively, having chosen the mvar x. Let $p > q$ (resp. $p = q$), and fix $0 \leq i \leq q$ (resp. $0 \leq i \leq p-1$). Define the *ith Sylvester-Habicht matrix*, denoted $\mathrm{SylvHa}_i(f, g)$, as the matrix whose rows are $x^{q-i-1}f, x^{q-i-2}f, \cdots, f, g, \cdots, x^{p-i-1}g$, considered as vectors in the basis $[x^{p+q-i-1}, \cdots, x, 1]$; it has $p+q-i$ columns and $p+q-2i$ rows. Then, the ith principal subresultant coefficient, denoted psc_i, is the determinant of the submatrix of $\mathrm{SylvHa}_i(f, g)$ obtained by taking the first $p + q - 2i$ columns. Then, the PSC set $\mathrm{PSC}(f, g) = \{\mathrm{psc}_i(f, g), 0 \leq i \leq \min(\deg(f), \deg(g)), \mathrm{psc}_i(f, g) \neq 0\}$.

Example 2.9. *Let* $f = 3x^2 + 5x + 6, g = 4x^2 + 2x + 1$*, with mvar* x*, and say we want* psc_0*. The Sylvester-Habicht matrix will have* 4 *columns and* 4 *rows. The basis for the rows is* $[x^3, x^2, x, 1]$*. For the first row, we compute* $x \cdot f = 3x^3 + 5x^2 + 6x$*, which is* $[3, 5, 6, 0]$ *in our basis. The second row is* $[0, 3, 5, 6]$*. For the third row, we take* g*, which yields* $[0, 4, 2, 1]$*. For the fourth row, we compute*

$x \cdot g$, yielding $[4,2,1,0]$ in our basis. Hence, we have the following matrix:

$$\begin{bmatrix} 3 & 5 & 6 & 0 \\ 0 & 3 & 5 & 6 \\ 0 & 4 & 2 & 1 \\ 4 & 2 & 1 & 0 \end{bmatrix}$$

Then, psc_0 is the determinant of the whole matrix, which is -343.

The operations defined above are used in the Hong projection operator. We let $\mathcal{D}$ represent the derivative operator. Then, the Hong projection operator $\mathrm{PROJH}(F)$ of a set F of polynomials is [Hong, 1990]:

$$\mathrm{PROJH}(F) = \mathrm{PROJ}_1(F) \cup \mathrm{PROJ}_2(F) \tag{2.11}$$

$$\mathrm{PROJ}_1(F) = \bigcup_{\substack{f \in F \\ f' \in \mathrm{RED}(f)}} [\{\mathrm{ldcf}(f')\} \cup \mathrm{PSC}(f', \mathcal{D}f')] \tag{2.12}$$

$$\mathrm{PROJ}_2(F) = \bigcup_{\substack{f,g \in F \\ f \prec g}} \bigcup_{f' \in \mathrm{RED}(f)} \mathrm{PSC}(f', g) \tag{2.13}$$

Above, $f \prec g$ denotes an arbitrary linear ordering, to not loop over redundant pairs. The operator PROJ_1 tells us to loop over the set F, and loop over the reducta set for each, and apply ldcf and PSC. The operator PROJ_2 tells us to loop over (non-redundant) pairs $f, g \in F$, take the first one's reducta set, and calculate the PSC set between each of them with g. Hong [1990] proved that this is a valid projection operator.

Now that we have defined the Hong projection operator, we demonstrate an example, using the sphere, as in Example 2.5.

Example 2.10. *Suppose we have the polynomial $x^2+y^2+z^2-1$. We want to compute the projection factors of this set, which only has a single polynomial. In keeping with our notation, we set $F_3 = \{x^2+y^2+z^2-1\}$. We now eliminate z and then y.*

1. *Eliminate z.*
   - *Apply* PROJ_1.
     - *Set $f = x^2+y^2+z^2-1$. Then, $\mathrm{RED}(f) = \{x^2+y^2+z^2-1, x^2+y^2-1\}$.*
       - *Set $f' = x^2+y^2+z^2-1$. We have $\mathrm{ldcf}(f') = 1$. Also, $\mathcal{D}f' = 2z$. We now compute* PSC.
         - *Set $i = 0$. SylvHa_0 is a 3×3 matrix, who's rows have the basis $[z^2, z, 1]$. We have:*

$$\mathrm{SylvHa}_0(f', \mathcal{D}f') = \begin{bmatrix} 1 & 0 & x^2+y^2-1 \\ 0 & 2 & 0 \\ 2 & 0 & 0 \end{bmatrix}$$

Then, $\mathrm{psc}_0(f', \mathcal{D}f')$ is the determinant of the entire above SylvHa *matrix, equaling $4-4x^2-4y^2$.*

· *Set* $i = 1$. SylvHa_1 *is a* 1×2 *matrix, who's rows have the basis* $[z, 1]$. *We have:*

$$\mathrm{SylvHa}_1(f', \mathcal{D}f') = \begin{bmatrix} 2 & 0 \end{bmatrix}$$

Then, $\mathrm{psc}_1(f', \mathcal{D}f')$ *is the determinant of the submatrix of the above matrix taking the first* 1 *columns, equaling* 2.

∗ *Set* $f' = x^2 + y^2 - 1$. *We have* $\mathrm{ldcf}(f') = x^2 + y^2 - 1$. *Also,* $\mathcal{D}f' = 0$. *We now compute* PSC. *As both are degree 0 in z, it is empty.*

- *Apply* PROJ_2*: can skip as we only have a single polynomial.*
- *Hence,* $F_2 = \{1, 2, x^2 + y^2 - 1, -4x^2 - 4y^2 + 4\}$. *We can drop the constants (see Section 2.4.2).*

2. *Eliminate y.*

- *Apply* PROJ_1.
  - *Set* $f = x^2 + y^2 - 1$. *Then,* $\mathrm{RED}(f) = \{x^2 + y^2 - 1, x^2 - 1\}$.
    - ∗ *Set* $f' = x^2 + y^2 - 1$. *We have* $\mathrm{ldcf}(f') = 1$. *Also,* $\mathcal{D}f' = 2y$. *We now compute* PSC. *We omit some details as we showed a detailed run-through above.*
      - · *Set* $i = 0$. *We have:*

$$\mathrm{SylvHa}_0(f', \mathcal{D}f') = \begin{bmatrix} 1 & 0 & x^2 - 1 \\ 0 & 2 & 0 \\ 2 & 0 & 0 \end{bmatrix}$$

Then, $\mathrm{psc}_0(f', \mathcal{D}f')$ *is the determinant of the entire above* SylvHa *matrix, equaling* $4 - 4x^2$.

· *Set* $i = 1$. *We have:*

$$\mathrm{SylvHa}_1(f', \mathcal{D}f') = \begin{bmatrix} 2 & 0 \end{bmatrix}$$

Take the first 1 *columns:* $\mathrm{psc}_1(f', \mathcal{D}f')$ *equals* 2.

∗ *Set* $f' = x^2 - 1$. *We have* $\mathrm{ldcf}(f') = x^2 - 1$. *Also,* $\mathcal{D}f' = 0$. *Lastly,* $\mathrm{PSC}(f', \mathcal{D}f') = \emptyset$.

  - *Set* $f = -4x^2 - 4y^2 + 4$. *Then,* $\mathrm{RED}(f) = \{-4x^2 - 4y^2 + 4, -4x^2 + 4\}$.
    - ∗ *Set* $f' = -4x^2 - 4y^2 + 4$. *We have* $\mathrm{ldcf}(f') = -4$. *Also,* $\mathcal{D}f' = -8y$. *We now compute* PSC.
      - · *Set* $i = 0$. *We have:*

$$\mathrm{SylvHa}_0(f', \mathcal{D}f') = \begin{bmatrix} -4 & 0 & 4 - 4x^2 \\ 0 & -8 & 0 \\ -8 & 0 & 0 \end{bmatrix}$$

Then, $\mathrm{psc}_0(f', \mathcal{D}f') = -256 + 265x^2$.

· *Set* $i = 1$. *We have:*

$$\mathrm{SylvHa}_1(f', \mathcal{D}f') = \begin{bmatrix} -8 & 0 \end{bmatrix}$$

Take the first 1 *columns:* $\mathrm{psc}_1(f', \mathcal{D}f') = -8$.

    * *Set* $f' = -4x^2 + 4$. *We have* $\text{ldcf}(f') = -4x^2 + 4$. *Also,* $\mathcal{D}f' = 0$. *Lastly,* $\text{PSC}(f', \mathcal{D}f') = \emptyset$.

- *Apply* PROJ_2.
  - *Set* $f = x^2 + y^2 - 1, g = -4x^2 - 4y^2 + 4$. *We have* $\text{RED}(f) = \{x^2 + y^2 - 1, x^2 - 1\}$.
    * *Set* $f' = x^2 + y^2 - 1$. *We have* $\text{psc}_0 = 0$ *and* $\text{psc}_1 = 0$ *(we omit the construction of the* SylvHa *matrices).*
    * *Set* $f' = x^2 - 1$. *We have* $\text{psc}_0 = 1 - 2x^2 + x^4$.
- *Hence,* $F_1 = \{-8, -4, 0, 1, 2, -256 + 256x^2, -4x^2 + 4, x^2 - 1, x^4 - 2x^2 + 1\}$. *We can drop the constants (see Section 2.4.2).*

Therefore, we have the projection factors after eliminating z: $F_2 = \{x^2 + y^2 - 1, -4x^2 - 4y^2 + 4\}$ *and the projection factors after eliminating y:* $F_1 = \{-256 + 256x^2, -4x^2 + 4, x^2 - 1, x^4 - 2x^2 + 1\}$.

2.4.2 Technical Lemmas About Reducta and Principal Subresultant Coefficients

We introduce several technical lemmas about reducta and principal subresultant coefficients through which we will study the behavior of the Hong projection, which will be especially useful for the polynomial systems we will study.

First, we have the following lemmas about reducta sets.

Lemma 2.11. *If f is degree* 0*, then* $\bigcup_{f' \in \text{RED}(f)} [\text{ldcf}(f') \cup \text{PSC}(f', \mathcal{D}f')] = \{f\}$.

Proof. $\text{RED}(f) = \{\text{red}^0(f)\} = f$, and $\text{ldcf}(f) = f$, and $\mathcal{D}f = 0$, so $\text{PSC}(f, \mathcal{D}f) = \emptyset$. Hence, we only have $\{f\}$. □

Lemma 2.12. *If f is degree* 1*, then* $\bigcup_{f' \in \text{RED}(f)} [\text{ldcf}(f') \cup \text{PSC}(f', \mathcal{D}f')] = \{\text{ldcf}(f), f - \text{ldt}(f)\}$.

Proof. Here, $\text{RED}(f) = \{f, f - \text{ldt}(f)\}$. Now, $\mathcal{D}f = \text{ldcf}(f)$ is degree-zero, so to compute $\text{PSC}(f, \mathcal{D}f) = \{\text{psc}_0(f, \mathcal{D}f)\}$, note that $\text{SylvHa}_0(f, \mathcal{D}f)$ is the 1×1 matrix with the entry ldcf f, so the determinant and hence $\text{psc}_0(f, \mathcal{D}f) = \text{ldcf}\, f$. Lastly, we will include $\text{ldcf}(f)$, and $\text{ldcf}(f - \text{ldt}(f)) = f - \text{ldt}(f)$, because it is degree-zero. So we are left with $\{\text{ldcf}(f), f - \text{ldt}(f)\}$. □

Next, we have the following lemmas about principal subresultant coefficients.

Lemma 2.13. *Let f, g, with g degree 0 and* $\deg(f) = d > 0$. *Then* $\text{PSC}(f, g) = \{(-1)^{\lfloor d/2 \rfloor} \cdot g^d\}$.

Proof. Here, $\text{PSC}(f, g) = \{\text{psc}_0(f, g)\}$. We form $\text{SylvHa}_0(f, g)$ as follows: it is a $d \times d$ matrix, and no rows will correspond to f, because $\deg(g) = 0$. So, it is the antidiagonal matrix with entries g. We perform $\lfloor d/2 \rfloor$ row swaps (i.e., first and last, second and second last, etc.) to make it a diagonal matrix, noting that every swap causes a sign change in the determinant. Then, the determinant of the diagonal matrix is g^d, which is multiplied by $(-1)^{\lfloor d/2 \rfloor}$ because of the swaps. □

Lemma 2.14. *Let f, g, with g degree 0, and f have a degree sequence, i.e., exponents of x with a non-zero coefficient, of D. Then,* $\bigcup_{f' \in \text{RED}(f)} \text{PSC}(f', g) = \{(-1)^{\lfloor d/2 \rfloor} \cdot g^d, d \in D\}$.

Proof. The degrees of the reducta set of f are exactly the degree sequence D. The result follows from Lemma 2.13. □

Remark 2.15. For the purposes of CAD, the sign change, though mathematically accurate, is not necessary to keep track of, because ultimately we care about the roots of the projection factors, which is unaffected by negation. Therefore, when we apply Lemmas 2.13 and 2.14, we will ignore the $(-1)^{\lfloor d/2 \rfloor}$ sign.

We summarize some key takeaways here:

- For PROJ_1, degree 0 polynomials (in the mvar, although they may still have positive degree in other variables), are just copied over into the projection factor set.
- For PROJ_2, we can ignore pairs with both degree-zero polynomials. If the pair has a single degree-zero polynomial, we can easily use Lemma 2.14.
- For constants, i.e., degree 0 in all variables, we need not store them, as they will be propagated to every subsequent projection factor set, and in the base phase do not contribute to creating any cells.

2.4.3 The CAD Algorithm

Having now defined the machinery of the Hong projection operator, and demonstrating some properties of it, we are now ready to give the algorithm of Collins [1974] below. We refer the reader to Chapters 5 and 11 of Basu et al. [2006] for a comprehensive description of the algorithm. Here, we break up CAD into two algorithms, which we call the *decision phase* and *solution formula phase*. The pseudocode of these algorithms is given in Algorithm 1 and Algorithm 2, respectively. The decision phase of CAD determines whether a polynomial system is consistent, whereas the solution formula phase constructs the full CAD tree as in Example 2.5.

The input of the decision phase is a set $F_k \subset \mathbb{R}[\{x_i\}_{i=1}^k]$ of polynomials in k variables. The algorithm has three main steps: projection, base case, and lifting. In the projection step, the set F_k is iteratively projected down to lower dimensions, forming sets of polynomials that are called *projection factors* in $\mathbb{R}^{k-1}, \mathbb{R}^{k-2}, \cdots \mathbb{R}^1$, denoted $F_{k-1}, F_{k-2}, \cdots F_1$, respectively. In the base case step, we form cells in $\mathbb{R}^1$ using F_1 and then store a single sample point for each cell. Finally, in the lifting step, the CAD is lifted iteratively back up to $\mathbb{R}_k$. The final result is a set of points in $\mathbb{R}^k$ where each point represents a cell over which each $f \in F_k$ is sign-invariant. Therefore, the truth of a system of polynomials where the atoms are composed of $f \in F_k$ can be determined by evaluation at each test point, due to the sign-invariance property. Thus, Algorithm 1 allows us to determine if *any* solution exists, which is sufficient for many applications, e.g., SMT.

If we want an algebraic description of *all* points that satisfy the system, like in the case of a sensitivity analysis, then we require Algorithm 2. This algorithm produces the desired result for *projection-definable* systems [Brown, 1999]. A system is projection-definable if no two cells that have different truth values share the same sign for any projection factor. In this case, the solution formula can be constructed purely using the signs of the projection factors. Algorithm 2 requires information from Algorithm 1, namely the set of all projection factors and cells (specifically, their sample points), and relies on the sign-invariance of projection factors over the cells in their respective dimension. Algorithm 2 is sufficient, because as we will show in Lemma 2.27, all polynomial systems that arise in our sensitivity analysis application are projection-definable. Systems that are not

Algorithm 1 Cylindrical algebraic decomposition: decision phase

```
Require: A set of polynomials $F_k \subset \mathbb{R}[\{x_i\}_{i=1}^{k}]$
Ensure: Sample points in each cell of the CAD adapted to $F_k$
    # Step 1: Projection
 1: for $i = k, k-1, \dots, 2$ do
 2:     From $F_i$, construct the set of projection factors $F_{i-1} \subset \mathbb{R}[\{x_j\}_{j=1}^{i-1}]$ by eliminating $x_i$
 3: end for
    # Step 2: Base Case
 4: $r \leftarrow$ Ordered list of roots of all $f \in F_1$, labeled $r_1, r_2, \dots$
 5: Construct cells: $\{r_j\}_{j=1}^{|r|} \cup \{(r_j, r_{j+1})\}_{j=1}^{|r|-1} \cup \{(-\infty, \min r), (\max r, \infty)\}$
    # Calculate sample points
 6: for each cell do
 7:     if cell is a point then
 8:         Sample point is the point itself
 9:     else if cell is an interval with finite endpoints then
10:         Sample point is average of endpoints
11:     else if cell is $(-\infty, \min r)$ then
12:         Sample point is $\min r - 1$
13:     else if cell is $(\max r, \infty)$ then
14:         Sample point is $\max r + 1$
15:     end if
16: end for
    # Step 3: Lifting
17: $\mathcal{C} \leftarrow$ sample points of cells in $\mathbb{R}^1$
18: for $i = 2, 3, \dots, k$ do
19:     $\mathcal{C}' \leftarrow \emptyset$
20:     for each cell $C \in \mathcal{C}$ do
21:         Evaluate the polynomials in $F_i$ at the sample point of $C$
22:         Find the roots of these polynomials and construct cells and sample points in $\mathbb{R}^i$ as in the
    base case
23:         Add these sample points to $\mathcal{C}'$
24:     end for
25:     $\mathcal{C} \leftarrow \mathcal{C}'$
26: end for
27: return $\mathcal{C}$
```

projection-definable are outside the scope of this thesis. The interested reader is referred to Brown [1999].

Algorithm 2 Cylindrical algebraic decomposition: solution formula phase

```
Require: A projection-definable polynomial system, and all projection factors $F_k \cup F_{k-1} \cup F_{k-2} \cup \cdots \cup F_1$ and the set of sample points $\mathcal{C}$ from Algorithm 1
Ensure: A CAD-based formula representing all the solutions to the polynomial system
 1: cell_formulas ← empty list
 2: for $C \in \mathcal{C}$ where the polynomial system holds over $C$ do
 3:     formulas ← empty list
 4:     for $f \in F_k \cup F_{k-1} \cup F_{k-2} \cdots \cup F_1$ do
 5:         if $f$ is negative at $C$ then
 6:             Append atom $f < 0$ to formulas
 7:         else if $f$ is zero at $C$ then
 8:             Append atom $f = 0$ to formulas
 9:         else if $f$ is positive at $C$ then
10:             Append atom $f > 0$ to formulas
11:         end if
12:     end for
13:     Append $\bigwedge$ formulas to cell_formulas
14: end for
15: return $\bigvee$ cell_formulas
```

The crucial step in Algorithm 1 is the projection step (line 2), which is now the most well-studied part of the algorithm. The first valid projection operator was given by Collins [1974] and later simplified by Hong [1990], which gives smaller projection factor sets while still maintaining soundness and completeness. In our implementation, we use the Hong [1990] projection operator, which was just defined in Section 2.4.1. In Algorithm 2, the conjunctions in line 13 can be visualized in the tree in Example 2.5 as the paths from the top of the tree to the leafs, while the disjunctions in line 15 are the set of all paths, which comprise the entire tree.

We now have all the information necessary to rigorously construct the CAD of an arbitrary system. As an example, we revisit the 3D sphere CAD and construct it per Algorithms 1 and 2.

Example 2.16. *For the unit sphere, our system is $x^2+y^2+z^2-1=0$, so $F_3 = \{x^2+y^2+z^2-1\}$. We eliminate z and then y.*

Algorithm 1:

- *Projection (see Example 2.10)*
  - *Eliminate z: $F_2 = \{x^2+y^2-1, -4x^2-4y^2+4\}$*
  - *Eliminate y: $F_1 = \{-256+256x^2, -4x^2+4, x^2-1, x^4-2x^2+1\}$*
- *Base case. The roots of F_1 are $\{-1, 1\}$. So the cells are $\{(-\infty,-1), -1, (-1,1), 1, (1,\infty)\}$. We choose the corresponding sample points $\{-2,-1,0,1,2\}$.*
- *Lifting*
  - *To $\mathbb{R}^2$:*

    - *Over sample point* -2*, we have the polynomials* $\{3+y^2, 12-4y^2\}$*, with roots* $\{-\sqrt{3}, +\sqrt{3}\}$*, hence cells* $\{(-\infty, -\sqrt{3}), -\sqrt{3}, (-\sqrt{3}, \sqrt{3}), \sqrt{3}, (\sqrt{3}, \infty)\}$*, with sample points* $\{-\sqrt{3}-1, -\sqrt{3}, 0, \sqrt{3}, \sqrt{3}+1\}$
    - *Over sample point* -1*, we have the polynomials* $\{y^2, -4y^2\}$*, with roots* $\{0\}$*, hence cells* $\{(-\infty, 0), 0, (0, \infty)\}$*, with sample points* $\{-1, 0, 1\}$
    - *Over sample point* 0*, we have the polynomials* $\{y^2-1, -4y^2+4\}$*, with roots* $\{-1, 1\}$*, hence cells* $\{(-\infty, -1), -1, (-1, 1), 1, (1, \infty)\}$*, with sample points* $\{-2, -1, 0, 1, 2\}$
    - *Over sample points* 1 *and* 2*, the analysis is the same as with sample point* -1 *and* -2*, respectively, because* x *always appears as* x^2 *in* F_1
  - *To* $\mathbb{R}^3$*: We plug in each* (x, y) *sample point into* $x^2+y^2+z^2-1$ *and construct cells and sample points for* z*; we omit the details for brevity. There are* 41 *cells in total.*

- *Truth of the system: Evaluate* $x^2+y^2+z^2-1$ *at each sample point and check if it equals zero. The system holds over the following* 6 *cells, represented by their* (x, y, z) *sample points:* $\{(-1,0,0), (0,-1,0), (0,0,-1), (0,0,1), (0,1,0), (1,0,0)\}$.

Algorithm 2:

The set of projection factors is $F_1 \cup F_2 \cup F_3$*. We evaluate their signs over each cell. We can ignore* $\{-4x^2-4y^2+4, -4x^2+4, -256+256x^2\}$ *as they are multiples of other projection factors. We can also ignore* x^4-2x^2+1*, because it equals* $(x^2-1)^2$*, so it is always non-negative, and zero if and only if* $-1+x^2$ *is zero, so it is also redundant to list in the solution formula.*[3] *Thus, the three projection factors that remain are* $\{x^2-1, x^2+y^2-1, x^2+y^2+z^2-1\}$*. This system is projection-definable, as no pair of cells with differing truth values share the same signs of the projection factors, which can be manually checked.*

Next, we show an example of constructing the solution formula for a given a cell. Over the cell with sample point $(0, 0, 1)$ *applied to the three remaining projection factors we get the atoms* $\{x^2-1<0, x^2+y^2-1<0, x^2+y^2+z^2-1=0\}$*, and hence the formula* $-1 < x < 1 \wedge -\sqrt{1-x^2} < y < \sqrt{1-x^2} \wedge z = \sqrt{1-x^2-y^2}$*. Following a similar approach for the other cells, we recover the full CAD of the sphere in Example 2.5.*

Although we have a general approach to construct the CAD of an arbitrary system, the key challenge is computational complexity. Due to the complicated ways polynomials can interact during the projection phase, the number of projection factors and the size of the CAD can quickly grow large. CAD has a complexity that depends polynomially on the degrees and number of polynomials, but is *doubly exponential* in the number of variables [England et al., 2015]. Indeed, it is possible to construct examples where this doubly exponential complexity is attained [Basu et al., 2006]. This presents a significant challenge in computing CADs for even more than a few variables. As well, the order of which variables to eliminate during the projection step can significantly affect both the runtime and the space needed to store the CAD.

[3]Note that these "algebraic reasoning" arguments like ignoring multiples, solving inequalities of univariate polynomials, etc., are not strictly part of CAD – they should be handled at the level of a (good) computer algebra system.

2.4.4 A Special Class of Polynomial System

In the remainder of this section we examine a specific class of polynomial systems that is motivated directly by our Markov reward process context. We demonstrate that due to the special structure of this system, its CAD can be generated much more efficiently than the general case. We also present a specialized version of the general CAD algorithm, tailored to this polynomial system class.

We consider a system of polynomials that have two different "types" of variables: x-type variables and α-type variables. We have η of the x-type variables, so that we index them $\{x_i\}_{i=1}^{\eta}$. The α-type variables are doubly indexed and are arranged as follows: there are ϕ simplices, and in the ith simplex, there are τ_i of the α-type variables. In other words, we index the α-type variables as $\{\alpha_{i,j}\}_{i\in[\phi],j\in[\tau_i]}$), where we define the notation $[m] := \{1,\dots,m\}$ for a positive integer m. By definition, each variable $\alpha_{i,j}$ is present in only a single simplex. Thus, there are a total of $\eta+\sum_{i=1}^{\phi}\tau_i$ variables. Let $\mathbb{R}[\{x_i\}_{i=1}^{k}, \{\alpha_{i,j}\}_{i\in[\phi],j\in[\tau_i]}]$ represent the ring of polynomials of these variables, and let f^* be a polynomial in this ring. We will call the atom $f^* \geq 0$, the *defining inequality* of the system.

Our polynomial system of interest is

$$\begin{aligned}
&f^* \geq 0 \\
&x_i \geq 0 \quad i \in [\eta] \\
&\sum_{j=1}^{\tau_i} \alpha_{i,j} = 1 \quad i \in [\phi] \\
&0 \leq \alpha_{i,j} \leq 1 \quad i \in [\phi], j \in [\tau_i]
\end{aligned} \tag{M}$$

We focus on the case where x_i is sign-constrained both because of practical motivations (e.g., rewards and costs associated with real-world problems tend to be sign-constrained) and because it leads to slightly more complex CAD trees (due to the presence of the endpoint 0) allowing us to showcase the full range of the CAD algorithm. The following development fully applies to the case where x_i is free, except that there will be fewer cells.

Remark 2.17. The connection to our sensitivity analysis motivation is that $f^* \geq 0$ is the condition the policymaker wants to test (e.g., NMB greater than a threshold), the x-type variables represents rewards, costs and benefits, and the α-type variables represent probabilities, i.e., $\boldsymbol{\pi}$ and the rows of $\mathbf{P}$.

Despite the general challenges described at the end of the previous subsection, we will show that we can construct a CAD for system (M) far more efficiently than general systems. This is made possible due to the following:

1. For a given i, the $\alpha_{i,j}$ variables form a unit simplex.

2. For a given i, j, the $\alpha_{i,j}$ variable only appears in a single simplex constraint.

3. The x-type variables only appear in $f^* \geq 0$.

Our approach takes advantage of this structure as follows. First, we construct a CAD of each simplex (which can be parallelized), and then conjunct them together. Then, relying on a special property associated with the projection factors of f^* that we describe later ("simplex-extensibility"), we can efficiently lift to include the x-type variables, which only show up in the defining inequality, to construct the full CAD.

CAD of simplices

Since all variables are restricted to be between 0 and 1, the construction of a simplex CAD is straightforward.

Theorem 2.18. *Consider a unit simplex $\{\alpha_j, j \in [\tau] \mid \sum_{j=1}^{\tau} \alpha_j = 1, 0 \leq \alpha_j \leq 1, j \in [\tau]\}$. Its CAD is given by the following tree:*

1. *The first level has three cells: $\alpha_1 = 0$, $0 < \alpha_1 < 1$, $\alpha_1 = 1$.*
2. *For $2 \leq i \leq \tau - 1$, for a given cell in level $i-1$, the children cells in level i are 0, $(0, 1 - \sum_{j=1}^{i-1} \alpha_j)$ and $1 - \sum_{j=1}^{i-1} \alpha_j$.*
3. *The last level ($i = \tau$) has a single child cell for each of the $\tau - 1$ level cells: $\alpha_\tau = 1 - \sum_{j=1}^{\tau-1} \alpha_j$.*

The tree also propagates down any cells defined by an equality on the preceding variables.

Proof. Observe that a valid representation of a simplex can be constructed as follows: $\alpha_1 \in [0, 1]$, $\alpha_2 \in [0, 1 - \alpha_1]$, $\alpha_3 \in [0, 1 - \alpha_1 - \alpha_2]$, and so on, until $\alpha_\tau = 1 - \sum_1^{\tau-1} \alpha_i$. Clearly, each variable is in $[0, 1]$. The sum $\sum_1^{\tau} \alpha_i \geq 1$, by summing the lower bounds of these intervals. Next, by looking at the partial sums of the upper bounds, we can see that $\alpha_1 + \alpha_2 \leq 1$, $\alpha_1 + \alpha_2 + \alpha_3 \leq 1$, and so on, so that $\sum_1^{\tau} \alpha_i \leq 1$. Therefore, $\sum_1^{\tau} \alpha_i = 1$, as required by the simplex constraint.

Then, we convert this into a valid CAD by decomposing each of the defining intervals into their cells, namely the endpoints and the open interval. □

Remark 2.19. By propagating any equality constraints active on the current cell, we may obtain a degenerate interval (i.e., a point) in which case there is only one new cell. This is the case if, for example, (taking a simplex with four variables) $0 < \alpha_1 < 1 \wedge \alpha_2 = 1 - \alpha_1$. Then, surely $\alpha_3 = \alpha_4 = 0$.

It is easy to extend this simplex construction to the case where $\sum_{i=1}^{\tau} \alpha_i = \kappa \leq 1$, as the defining representation of the simplex is now $\alpha_1 \in [0, 1]$, $\alpha_2 \in [0, \kappa - \alpha_1]$, $\alpha_3 \in [0, \kappa - \alpha_1 - \alpha_2]$, and so on, until $\alpha_\tau = \kappa - \sum_{i=1}^{\tau-1} \alpha_i$. Also, we can extend this construction to the inequality case, where $\sum_{i=1}^{\tau} \alpha_i \leq \kappa \leq 1$, by replacing the node at the final level with $\alpha_\tau \leq \kappa - \sum_{i=1}^{\tau-1} \alpha_i$.

Since each α-type variable only occurs in a single simplex, it is trivial to create a CAD for a set of simplices, by "gluing" them together.

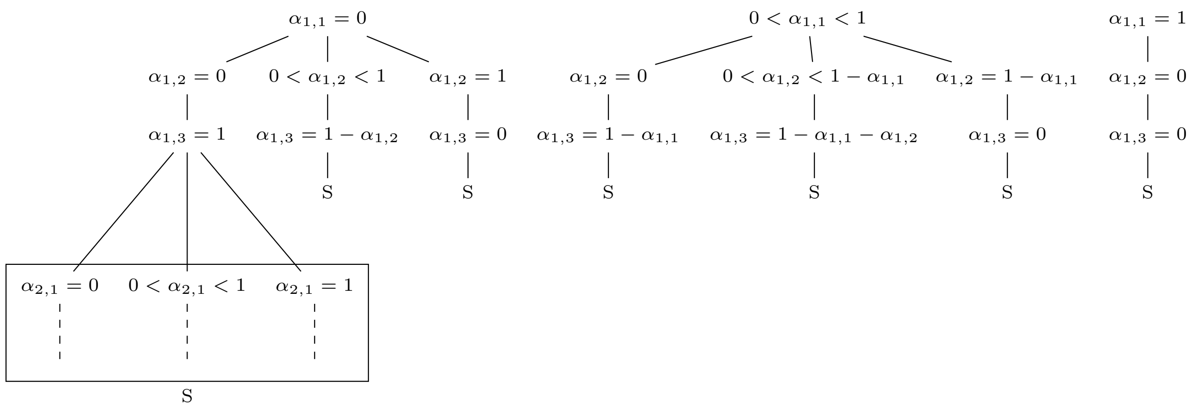
Corollary 2.20. *Consider ϕ unit simplices $\{\alpha_{i,j}, i \in [\phi], j \in [\tau_i] \mid \sum_{j=1}^{\tau_i} = 1, 0 \leq \alpha_{i,j} \leq 1, i \in [\phi], j \in [\tau_i]\}$. To construct a cell in the CAD of the conjunction of these simplices, choose a single cell from each of the individual simplices' CADs, and conjunct them. The full CAD is the disjunction of all such cells.*

Proof. We proceed by induction. In the base case, with one simplex, it is trivially true. For the inductive step, assume we have the CAD of the conjunction of m simplices. Then, to lift this CAD to include the $m + 1$-st simplex, first note that because the α-type variables only appear in a single simplex, the lifting is identical over each cell, and indeed the lifting process will simply yield the additional $m + 1$-st simplex itself. Therefore, each cell in the CAD of the conjunction of $m + 1$

simplices is the conjunction of a single cell from each of the simplices individually, and the full CAD is the disjunction over all such cells, as needed.

Note that this also follows from the distributivity of conjunction over disjunction for an arbitrarily indexed family of sets: e.g., see Monk [1969, Theorem 5.21]. □

Example 2.21. *Consider two simplices* $\alpha_{1,1}+\alpha_{1,2}+\alpha_{1,3}=1$ *and* $\alpha_{2,1}+\alpha_{2,2}+\alpha_{2,3}=1$. *The CAD of their conjunction is:*



where S represents the CAD of the second simplex, $\alpha_{2,1}+\alpha_{2,2}+\alpha_{2,3}=1$, *shown partially on the leftmost branch, which has identical structure to the CAD of the first simplex.*

The intuition of the preceding corollary is that because each simplex is independent, we can simply copy and conjunct them together. For example, if we have two simplices, take each cell in the first simplex, and attach a copy of the second simplex. This leads to a bound on the size of the CAD of simplices.

Corollary 2.22. *The number of cells in a CAD of the conjunction of* ϕ *simplices, where the* i*th simplex has* τ_i *variables, is* $O(3^{\sum_{i=1}^{\phi}\tau_i}) = O(3^{\phi \max_i \tau_i})$.

Proof. The CAD of a single simplex has at most 3 children at each level, except at the last level where it definitely has one. So the number of cells for the ith simplex is $O(3^{\tau_i-1}) = O(3^{\tau_i})$. Then, if we conjunct ϕ simplices together, we multiply the number of cells: $O(3^{\tau_1}\times 3^{\tau_2}\cdots\times 3^{\tau_\phi}) = O(3^{\sum_{i=1}^{\phi}\tau_i})$. Since each $\tau_i \le \max_{1\le j\le\phi}\tau_j$, we also have the bound $O(3^{\phi\cdot\max_j \tau_j})$. □

In general, the size of a CAD is doubly exponential in the number of variables, whereas for special case of simplices it is singly exponential: the savings are due to the geometry of the simplex and the assumption of their disjointness.

Simplex-extensibility: Lifting the simplex CAD to include f^*

The previous section showed that it was easy to construct a CAD for a set of simplices. Now, we wish to lift the CAD to include f^*. The method of incremental CAD, which we discussed in the literature review, applies here. In the general case, it requires significant computation, due to the many possibilities of how CADs can interact. Therefore, we focus on characterizing conditions on f^* such that extending the simplex CAD is easy.

Suppose we repeatedly apply a projection operator to the set F of polynomials that make up our system (M) to eliminate all x-type variables. Then, we are left with a projection factor set F', where each $f \in F'$ is a polynomial in the α-type variables. If the CAD of the simplex constraints is *also* the CAD of F', then we can simply stop the projection phase here and begin the lifting procedure from the simplex CAD to construct a CAD for the full system. The value of such an approach is the avoidance of calculating subsequent projection factor sets, which can blow up quickly. Thus, we wish to characterize the polynomials f^* such that this is true.

To do so, we will now use the technical lemmas that we introduced in Chapter 2.4.2 which will help us to apply the Hong projection operator to instances of system (M).

Below, we provide a motivating example that will demonstrate the situation we discussed above.

Example 2.23. *Consider the system $\alpha_1 x_1 + \alpha_2 x_2 - 1 \geq 0$ and $\alpha_1 + \alpha_2 = 1$. The set of polynomials whose atoms form the system are $\{\alpha_1 x_1 + \alpha_2 x_2 - 1, \alpha_1 + \alpha_2 - 1\}$. Iteratively applying the Hong projection:*

- *Eliminate x_1: $\{\alpha_1, \alpha_1 x_2 - 1, \alpha_1 + \alpha_2 - 1\}$*
- *Eliminate x_2: $\{\alpha_1, \alpha_2, \alpha_1 + \alpha_2 - 1\}$*

Now, observe that the final projection factor set has the same CAD as the CAD of the simplex associated with $\alpha_1 + \alpha_2 = 1$, so we can lift from it. Specifically, observe that each of the elements of the projection factor set are sign-invariant in each of the cells.

The sign-invariance property observed in the previous example is crucial to our development. We call this property *simplex-extensible.*

Definition 2.24 (Simplex-extensible)**.** An instance of the system (M) is *simplex-extensible* if the set of projection factors after eliminating all x-type variables is sign-invariant over each cell in the CAD of the conjunction of the respective simplex constraints.

Example 2.25. *To demonstrate an instance that is not simplex-extensible, take $f^* = x_1(\alpha_2^2 - \alpha_1^2) + x_2\alpha_2 - 1$, with the simplex constraint $\alpha_1 + \alpha_2 - 1 = 0$. If we calculate the projection factor set after eliminating x_1 and x_2, we obtain $\{-1 + \alpha_2, \alpha_2^2, \alpha_1 + \alpha_2 - 1, \alpha_2^2 - \alpha_1^2\}$ (we omit the details for brevity). The last polynomial, $\alpha_2^2 - \alpha_1^2$, is not sign-invariant on the simplex CAD. For example, take $(\alpha_1, \alpha_2) = (0.2, 0.8)$, which satisfies the simplex: at this point, the polynomial is positive. Now take $(\alpha_1, \alpha_2) = (0.8, 0.2)$, which also satisfies the simplex and indeed also lies in the same cell in the simplex CAD (namely the cell $0 < \alpha_1 < 1 \wedge \alpha_2 = 1 - \alpha_1$): at this point, the polynomial is negative. Therefore, it is not sign-invariant within a cell of the simplex CAD.*

Characterizing the full class of simplex-extensible systems is difficult, as by definition it requires the repeated application of the Hong projector until all x-type variables are eliminated. However, using the series of technical lemmas presented in Section 2.4.2, we can derive several special cases where the characterization is much easier. An important class of polynomials is of the form $f^* = g_0 + \sum_{i=1}^{\eta} x_i g_i$, where each function $\{g_i\}_{i=0}^{\eta}$ is a function of the α-type variables, for which we can easily check its simplex-extensibility.

Theorem 2.26. *Let $f^* = g_0 + \sum_{i=1}^{\eta} x_i g_i$, where each function $\{g_i\}_{i=0}^{\eta}$ is a polynomial of the α-type variables. Then, this system is simplex-extensible if and only if each of $\{g_i\}_{i=0}^{\eta}$ is sign-invariant over the CAD of the simplex constraints.*

Proof. We eliminate the x-type variables in turn. First, if we set x_η as the mvar, then for PROJ$_1$, we copy over the simplex constraints, and for f^*, we keep the ldcf (which is g_η) and subtract the ldt (which is $x_\eta g_\eta$). For PROJ$_2$, we raise the simplex constraints to the power of one, but we already have them. So, this yields a projection factor set of $\{g_\eta, g_0 + \sum_1^{\eta-1} x_i g_i\}$ plus the simplex constraints.

We then similarly eliminate $x_{\eta-1}$, which yields the projection factor set $\{g_\eta, g_{\eta-1}, g_0 + \sum_1^{\eta-2} x_i g_i\}$ plus the simplex constraints. Inductively, after eliminating all x-type variables, we are left with $\{g_i\}_0^\eta$ plus the simplex constraints.

By definition, the instance is simplex-extensible if and only if each of these projection factors is sign-invariant over the simplex CAD. Trivially, the simplex constraints are sign-invariant over the simplex. Therefore, it is simplex-extensible if and only if each of $\{g_i\}_0^\eta$ are sign-invariant over the simplex CAD. □

Theorem 2.26 captures a general class of f^*, in particular covering the relevant functions for our Markov reward process context. More general classes of functions that exhibit simplex-extensibility are provided in Section 2.4.4.

In Example 2.23, a system with $f^* = x_1\alpha_1 + x_2\alpha_2 - 1$ is simplex-extensible by Theorem 2.26. Another example is that an instance with $f^* = x_1\alpha_1^2 + x_2\alpha_2^2 - 1$ is simplex-extensible, because α_1^2 and α_2^2 are strictly positive except when $\alpha_1 = 0$ (or likewise for $\alpha_2 = 0$), but that is its own cell. In Example 2.25, since f^* doesn't satisfy the property in Theorem 2.26, the system is not simplex-extensible.

Thus, with a suitably chosen f^*, system (M) is simplex-extensible, which means we can stop projecting once we have eliminated all x-type variables and begin the lifting phase from the simplex CAD. Namely, we take sample points from the simplex CAD cells, plug them into the projection factors, calculate roots, and then lift on the next x-type variable. This now gives us a set of sample points over which we can evaluate the consistency of the system. We discuss the construction of the solution formula next.

Solution formula construction

Recall that we can use Algorithm 2 to generate the solution formula for systems that are projection-definable. Next, we show that our system of interest is projection-definable for a suitably chosen f^*.

Lemma 2.27. *Assume we have an instance of system* (M) *that is simplex-extensible, with* $f^* = g_0 + \sum_{i=1}^{\eta} x_i g_i$*, where each function* $\{g_i\}_{i=0}^{\eta}$ *is a polynomial of the* α*-type variables. Then, this system is projection-definable.*

Proof. First, we already have the solution formula for the simplex CAD, so we need only focus on the solution formula when lifting to the x-type variables. The set of all projection factors, following the proof of Theorem 2.26, is:

$$\left\{g_\eta, g_0 + \sum_1^{\eta-1} x_i g_i\right\} \bigcup \left\{g_\eta, g_{\eta-1}, g_0 + \sum_1^{\eta-2} x_i g_i\right\} \bigcup \cdots \bigcup \{g_i\}_0^\eta$$
$$= \{g_i\}_0^\eta \bigcup \left\{ g_0 + \sum_{j=1}^{\eta-i} x_i g_i \right\}_{i=1}^{\eta-1}$$

We can construct a solution formula from the projection factors as we discussed previously, by conjuncting atoms in accordance with the signs of these projection factors. As $\{g_i\}_0^\eta$ are sign-invariant over each cell, due to simplex-extensibility, it is redundant to include them. Secondly, atoms formed from the following set:

$$\left\{ g_0 + \sum_{j=1}^{\eta-i} x_i g_i \right\}_{i=1}^{\eta-1}$$

are linear in the respective x_i, hence have at most one root over the cell that we are lifting from, so that the sign of the projection factor uniquely determines a region. Therefore, this system is projection-definable. □

Given Lemma 2.27, we can specialize Algorithm 2 to system (M). We can also specialize Algorithm 1 with knowledge of the specific projection factors associated with the simplex constraints and the structure of f^*. Algorithm 3, combines the specialized versions of Algorithms 1 and 2, outputting the solution formula for any instance of (M) with $f^* = g_0 + \sum_{i=1}^{\eta} x_i g_i$.

Algorithm 3 Constructing the CAD solution formula of an instance of system (M) with $f^* = g_0 + \sum_{i=1}^{\eta} x_i g_i$

```
Require: A simplex-extensible instance of system (M) with $f^* = g_0 + \sum_{i=1}^{\eta} x_i g_i$
Ensure: A CAD solution formula
 1: for each simplex constraint in the instance of system (M) do
 2:     Construct the CAD according to Theorem 2.18, storing sample points
 3: end for
 4: cells ← conjunction of the simplex CADs according to Corollary 2.20
 5: for $x_i$ in $\{x_i\}_{i=1}^{\eta}$ do
 6:     cells_new ← cells
 7:     for each cell in cells do
 8:         sample ← sample point of cell
 9:         if $g_i = 0$, evaluated at sample then
10:             Add the cell $x_i \geq 0$ as a child of cell with a sample point, in cells_new
11:         else
12:             $r_i \leftarrow -(g_0 - \sum_{j=1}^{i-1} x_j g_j)/g_i$ evaluated at sample
13:             if $r_i \leq 0$ then
14:                 Add the cell $x_i \geq 0$ as a child of cell, with a sample point, to cells_new
15:             else
16:                 Add cells $x_i = 0, 0 < x_i < r_i, x_i = r_i$, and $x_i > r_i$ as children of cell, with their
    sample points, in cells_new
17:             end if
18:         end if
19:     end for
20:     cells ← cells_new
21: end for
22: for each cell in cells do
23:     Evaluate the system using its sample point
24:     if the instance of system (M) does not hold then
25:         Delete cell
26:     end if
27: end for
28: return cells
```

Line 12 of Algorithm 3 is the key step that leverages the structure of f^* to construct the CAD efficiently. At the level of a particular x_i, there is only one projection factor, which has a unique root (see proof of Theorem 2.26):

$$r_i = \frac{-g_0 - \sum_{j=1}^{i-1} x_j g_j}{g_i}.$$

The result is that there will be no more than four new cells created at each level of the x-type variables (line 16). This observation directly leads to a bound on the size of the CAD tree, and the result that the tree is smaller than the size of a CAD of a general polynomial system with the same number of variables and constraints.

Corollary 2.28. *Assume we have an instance of system* (M) *that is simplex-extensible, with $f^* = g_0 + \sum_{i=1}^{\eta} x_i g_i$, where each function $\{g_i\}_{i=0}^{\eta}$ is a polynomial of the α-type variables. Then, the number of cells in its CAD is $O(3^{\sum_{i=1}^{\phi} \tau_i} \times 4^{\eta})$.*

Proof. The number of cells of the simplex CAD is as per Corollary 2.22. After the simplex CAD, each level has at most 4 cells, so 4^{η} cells for the x-type variables over each cell of the simplex CAD. □

Corollary 2.29. *Let the number of cells of an instance of system* (M)*, under the assumptions of Corollary 2.28, be N_M, and the number of cells in the CAD of a general polynomial system with the same number of variables and constraints be N. Then, $N_M = o(N)$.*

Proof. The number of variables in our system is $\eta + \sum_1^{\phi} \tau_i$, the number of polynomials is $\phi + 1$ (the ϕ simplices plus f^*). We let the maximum degree (over all variables, over all polynomials) be denoted as d: note that in this corollary we assume that f^* is linear in the x-type variables, but we make no assumptions on the degree of the α-type variables, so we simply denote it as d. Based on the complexity analysis in England et al. [2015], the dominant term of the bound of the number of cells of a general CAD, N, substituting our values, is:

$$(2d)^{2^{\eta+\sum_1^{\phi}\tau_i}-1}(\phi+1)^{2^{\eta+\sum_1^{\phi}\tau_i}-1}2^{2^{\eta+\sum_1^{\phi}\tau_i-1}-1}$$

Whereas when solving with the methods we have developed, we have the bound on N_M:

$$3^{\sum_{i=1}^{\phi}\tau_i} \times 4^{\eta} < 4^{\eta+\sum_{i=1}^{\phi}\tau_i}$$

For simplicity of notation, let $\psi = \eta + \sum_{i=1}^{\phi} \tau_i$. Then, we will prove that $4^{\psi} = o(2^{2^{\psi-1}-1})$. Consider the following ratio:

$$\frac{4^{\psi}}{2^{2^{\psi-1}-1}} = \frac{2^{2\psi}}{2^{2^{\psi-1}-1}} = 2^{2\psi - 2^{\psi-1}+1}$$

As $\psi \to \infty$, the exponent becomes large and negative, so that the term itself goes to 0. So, $4^{\psi} = o(2^{2^{\psi-1}-1})$. Hence, by properties of little-o, $N_M = o(N)$. □

More concretely, while the size of a general CAD is doubly exponential in size, the size of this CAD is only *singly exponential.* While this is a significant reduction theoretically, we may still consider this complexity to be too high for practical purposes. However, any tree data structure with a constant number of children for each node will have a total number of nodes that is singly exponential in the number of levels of the tree. Lastly, we show that even deciding feasibility of our system (let alone enumerating the full tree), even in the simplex-extensible case, is NP-hard. The proof is via reduction from 3-SAT.

Theorem 2.30. *Consider the decision problem of deciding the feasibility of an instance of system* (M) *that is simplex-extensible, with $f^* = g_0 + \sum_{i=1}^{\eta} x_i g_i$, where each function $\{g_i\}_{i=0}^{\eta}$ is a polynomial of the α-type variables. This problem is NP-hard. Hence, deciding the feasibility of a general instance of system* (M) *is NP-hard.*

Proof. Suppose we have an instance of 3-SAT, i.e., Boolean satisfiability with three literals in each clause, with m clauses. We refer the reader to Papadimitriou and Steiglitz [2013] for full details on the definition of 3-SAT. We first exhibit a reduction to a simplex-extensible instance of system (M) with $f^* = g_0 + \sum_{i=1}^{\eta} g_i$.

For each Boolean variable z_i the 3-SAT instance, introduce variables $\alpha_{i,1}$ (representing z_i being true) and $\alpha_{i,2}$ (representing z_i being false), with the constraints that $0 \leq \alpha_{i,1}, \alpha_{i,2} \leq 1$ and $\alpha_{i,1} + \alpha_{i,2} = 1$. Note that this means that in the context of system (M), we have ϕ equaling the number of Boolean variables, and $\tau = 2$ for each simplex. With these α variables, form the polynomial $f_1 = \sum_{i=1}^{\phi} \sum_{j=1}^{2} \alpha_{i,j}(1 - \alpha_{i,j})$. Observe that $f_1 = 0$ if and only if all of the α-type variables are either 0 or 1.

Next, take a clause $C_i = (\ell_{i,1}, \ell_{i,2}, \ell_{i,3})$ in the 3-SAT instance. Define the term $I_{i,j}$ which is $1 - \alpha_{k,1}$ if $\ell_{i,j} = z_k$ or $\alpha_{k,1}$ if $\ell_{i,j} = \neg z_k$. So, for each clause C_i, form the product $I_{i,1} I_{i,2} I_{i,3}$, which is clearly 0 if at least one of the constituent α variables is 0, i.e., if at least one of the respective z variables is 1 – in other words, it is 0 if and only if the clause C_i is true. Then, we take the sum over all clauses, forming the polynomial $f_2 = \sum_{i=1}^{m} I_{i,1} I_{i,2} I_{i,3}$. Hence, $f_2 = 0$ if and only if all clauses are true.

Finally, form the following polynomial inequality:

$$f^* := -\left(\underbrace{\sum_{i=1}^{\phi} \sum_{j=1}^{2} \alpha_{i,j}}_{f_1} + \underbrace{\sum_{i=1}^{m} I_{i,1} I_{i,2} I_{i,3}}_{f_2} \right) \geq 0$$

We now have an instance of system (M). We may observe that there are no x-type variables here: we can vacuously add them to f^* as, e.g., by adding $x - x$; this will have no effect on our construction. All the α variables are continuous between 0 and 1, and lie on disjoint simplices. This f^* is indeed in the form $g_0 + \sum_{i=1}^{0} g_i x_i$. As there are no x terms, we can simply set $g_0 = f^*$. Furthermore, as both f_1 and f_2 are non-negative, then g_0 is sign-invariant. Hence, this system is simplex-extensible.

We now have to show that this reduction is valid. First, suppose we have a satisfying assignment of the z variables to the 3-SAT instance. This implies that all of the α variables are either 0 or 1, by construction, so $f_1 = 0$, and due to all clauses being true we have $f_2 = 0$. So, $f^* = -(0+0) = 0 \geq 0$. So, the instance of system (M) is feasible.

Secondly, suppose we have a feasible solution to the instance of system (M). Since $f_1, f_2 \geq 0$ always, we have that $f^* \leq 0$ always. So, with a feasible solution, we have $f^* \leq 0$ and $f^* \geq 0$, so

$f^* = 0$. This implies that both $f_1 = 0$ and $f_2 = 0$. If $f_1 = 0$, all the α variables are either 0 or 1, meaning we can form valid Boolean variables z from them. Secondly, if $f_2 = 0$, we also have a satisfying assignment to the 3-SAT instance.

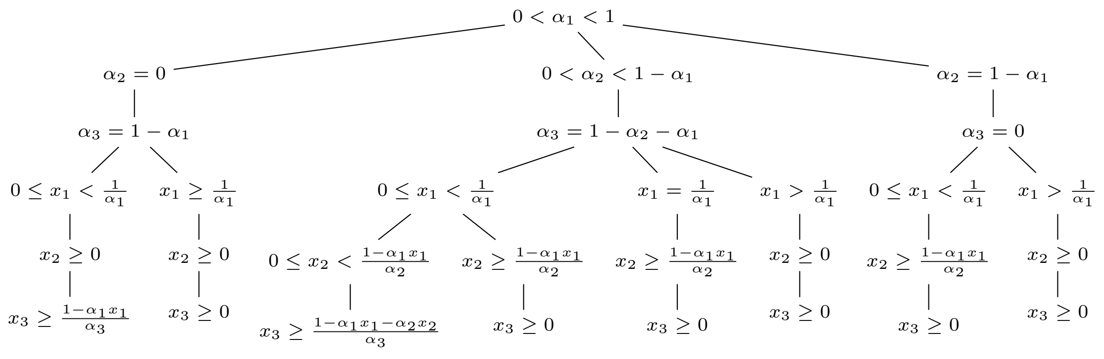
Hence, the constructed instance of (M) is feasible if and only if the original instance of 3-SAT is satisfiable. The NP-hardness of simplex-extensible (M) with $f^* = g_0 + \sum_{i=1}^{\eta} g_i$ and the NP-hardness of general (M) follow. □

Remark 2.31. Note that the construction of f_2 above uses very common techniques that are used to show complexity of polynomial systems. In an arbitrary polynomial system, we can fix the α variables to be binary by introducing the constraint $\alpha(\alpha - 1) = 0$. However, we cannot introduce additional constraints when forming system (M). We get around this by incorporating the constraints implicitly into f^*, so that the α still end up being constrained to be binary.

Remark 2.32 (Unsigned x-type variables). Algorithm 3 applies to the case where the x-type variables are free, with slight modifications to lines 10, 14, and 16. Theorem 2.26 and Lemma 2.27 also hold because the sign of x does not affect the sign-invariance of the g_i functions, since the latter are functions of only the α-type variables.

The following example illustrates a CAD tree built using Algorithm 3.

Example 2.33. *Let* $f^* = \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 - 1$*, with* $f^* \geq 0$*,* $\alpha_1 + \alpha_2 + \alpha_3 = 1$*,* $0 \leq \alpha_i \leq 1$*, and* $x_i \geq 0$ *for* $i = 1, 2, 3$*. A partial CAD (only over the cell* $0 < \alpha_1 < 1$*, for brevity) is given below.*


$0 < \alpha_1 < 1$
$\alpha_2 = 0$
$0 < \alpha_2 < 1 - \alpha_1$
$\alpha_2 = 1 - \alpha_1$
$\alpha_3 = 1 - \alpha_1$
$\alpha_3 = 1 - \alpha_2 - \alpha_1$
$\alpha_3 = 0$
$0 \leq x_1 < \frac{1}{\alpha_1}$
$x_1 \geq \frac{1}{\alpha_1}$
$0 \leq x_1 < \frac{1}{\alpha_1}$
$x_1 = \frac{1}{\alpha_1}$
$x_1 > \frac{1}{\alpha_1}$
$0 \leq x_1 < \frac{1}{\alpha_1}$
$x_1 > \frac{1}{\alpha_1}$
$x_2 \geq 0$
$x_2 \geq 0$
$0 \leq x_2 < \frac{1-\alpha_1 x_1}{\alpha_2}$
$x_2 \geq \frac{1-\alpha_1 x_1}{\alpha_2}$
$x_2 \geq \frac{1-\alpha_1 x_1}{\alpha_2}$
$x_2 \geq 0$
$x_2 \geq \frac{1-\alpha_1 x_1}{\alpha_2}$
$x_2 \geq 0$
$x_3 \geq \frac{1-\alpha_1 x_1}{\alpha_3}$
$x_3 \geq 0$
$x_3 \geq \frac{1-\alpha_1 x_1 - \alpha_2 x_2}{\alpha_3}$
$x_3 \geq 0$
$x_3 \geq 0$
$x_3 \geq 0$
$x_3 \geq 0$
$x_3 \geq 0$

Extensions: Increasing failure rate

The *increasing failure rate* (IFR) property is an important property of transition probability matrices in many healthcare and engineering applications [Barlow and Proschan, 1996]. It captures the notion that in worse health states, a patient (or machine) is more likely to degrade. More precisely, the rows of the transition probability matrix are in increasing stochastic order. We consider a Markov chain with ϕ states, so ϕ unit simplices, each with ϕ variables: $\{\alpha_{i,j}, i \in [\phi], j \in [\phi] \mid \sum_{j=1}^{\phi} \alpha_{i,j} = 1, 0 \leq \alpha_{i,j} \leq 1, i \in [\phi], j \in [\phi]\}$.

Definition 2.34 (Increasing failure rate (IFR) [Alagoz et al., 2007])**.** Consider ϕ unit simplices as above. This set of simplices is said to be IFR if, for each $h \in [\phi]$, we have that $\sum_{\ell=h}^{\phi} \alpha_{i,\ell}$, as a function of i, is nondecreasing in i, where $i \in [\phi]$.

The IFR condition introduces a series of linear inequalities as constraints on the simplices. These inequalities span across multiple simplices, so the constraints are not disjoint in the simplex variables anymore. Thus, we cannot use the prior argument to conjunct them to construct the CAD for the set of simplices. However, we can still efficiently construct a CAD for a set of simplices with the IFR property. Our strategy will be to construct the full CAD in the usual variable ordering: $\alpha_{1,1}, \ldots, \alpha_{1,\phi}, \alpha_{2,1}, \ldots, \alpha_{2,\phi}, \ldots, \alpha_{\phi,1}, \ldots \alpha_{\phi,\phi}$. By manipulating the inequalities in the IFR definition, we can characterize each α variable in terms of the preceding variables in this ordering.

Theorem 2.35. *Consider a set of ϕ unit simplices, each with ϕ states, $\{\alpha_{i,j}, i \in [\phi], j \in [\phi] \mid \sum_{j=1}^{\phi} \alpha_{i,j} = 1, 0 \leq \alpha_{i,j} \leq 1, i \in [\phi], j \in [\phi]\}$. Also, fix the variable ordering $\alpha_{1,1}, \alpha_{1,2}, \alpha_{1,3}, \cdots, \alpha_{1,\phi}, \alpha_{2,1}, \cdots, \alpha_{2,\phi}, \cdots \alpha_{\phi,\phi}$. If this set of simplices is IFR, their CAD is:*

- *For each i, the cells for $\alpha_{i,j}$, for $j < \phi$ are the cells $0, (0, \sum_{\ell=1}^{j} \alpha_{i-1,\ell} - \sum_{\ell=1}^{j-1} \alpha_{i,\ell}), \sum_{\ell=1}^{j} \alpha_{i-1,\ell} - \sum_{\ell=1}^{j-1} \alpha_{i,\ell}$.*
- *The singular cell for $\alpha_{i,\phi}$ for each i is $\{1 - \sum_{\ell=1}^{\phi-1} \alpha_{i,\ell}\}$.*
- *If $i = 1$, then we replace $\sum_{\ell=1}^{j} \alpha_{i-1,\ell}$ with 1.*
- *The tree propagates any equality constraints on preceding variables in the current cell.*

Proof. First, from the definition of IFR, we can write that, for a given $h \in [1, \phi]$, we have:

$$\begin{aligned}
&\sum_{\ell=h}^{\phi} \alpha_{1,\ell} \leq \sum_{\ell=h}^{\phi} \alpha_{2,\ell} \leq \cdots \leq \sum_{\ell=h}^{\phi} \alpha_{\phi,\ell} \\
\iff\ & 1 - \sum_{\ell=1}^{h-1} \alpha_{1,\ell} \leq 1 - \sum_{\ell=1}^{h-1} \alpha_{2,\ell} \leq \cdots \leq 1 - \sum_{\ell=1}^{h-1} \alpha_{\phi,\ell} \\
\iff\ & \sum_{\ell=1}^{h-1} \alpha_{1,\ell} \geq \sum_{\ell=1}^{h-1} \alpha_{2,\ell} \geq \cdots \geq \sum_{\ell=1}^{h-1} \alpha_{\phi,\ell}
\end{aligned}$$

Note that when $h = 1$, the inequalities are just $1 \leq \cdots \leq 1$, which adds no information.

For the $\alpha_{i,j}$ variables for $i = 1$, we have the usual characterization that $\alpha_{1,1} \in [0, 1]$, $\alpha_{1,2} \in [0, 1 - \alpha_{1,1}]$, and so on, until $\alpha_{1,\phi} = 1 - \alpha_{1,\phi-1} - \alpha_{1,\phi-2} - \cdots - \alpha_{1,1}$.

For $\alpha_{i,j}$, $i > 1$, from the inequalities in the IFR definition when $h = j + 1$, we get that:

$$\begin{aligned}
&\sum_{\ell=1}^{j} \alpha_{i-1,\ell} \geq \sum_{\ell=1}^{j} \alpha_{i,\ell} \\
\iff\ & \alpha_{i,j} \leq \sum_{\ell=1}^{j} \alpha_{i-1,\ell} - \sum_{\ell=1}^{j-1} \alpha_{i,\ell}
\end{aligned}$$

Recall that the usual bound $\alpha_{i,j} \leq 1 - \sum_{\ell=1}^{j-1} \alpha_{i,\ell}$. Since $\sum_{\ell=1}^{j} \alpha_{i-1,\ell} \leq 1$, the above is a stricter bound than we would have without the IFR constraint. Also, this provides an exact characterization of $\alpha_{i,j}$ in terms of the α-variables that precede it in our variable ordering.

Lastly, as implied by row-stochasticity, we must have $\alpha_{i,\phi} = 1 - \sum_{\ell=1}^{\phi-1} \alpha_{i,\ell}$. The CAD decomposition follows. □

Remark 2.36. The intuition in the bounds on α variables implied by the CAD is that if we write out the α variables in a matrix, a given $\alpha_{i,j}$ is bounded from above by the sum of the α variables in the preceding row (row $i-1$) up to the jth column, minus the sum of the α variables in the same row (row i) up to the $j-1$th column. So, $\alpha_{i,j}$ is larger if the probabilities in the row above it and to the left are larger, in order to fulfill IFR, and is smaller if the probabilities in the same row but precede it are larger, in order to fulfill row-stochasticity.

Example 2.37. *Suppose we have two simplices $\alpha_{1,1}+\alpha_{1,2}=1$ and $\alpha_{2,1}+\alpha_{2,2}=1$, and they are IFR. Its CAD is:*

$$\begin{tikzcd}
\alpha_{1,1}=0 \arrow[d, dash] & 0<\alpha_{1,1}<1 \arrow[d, dash] & \alpha_{1,1}=1 \arrow[d, dash] \\
\alpha_{1,2}=1 \arrow[d, dash] & \alpha_{1,2}=1-\alpha_{1,1} \arrow[d, dash] & \alpha_{1,2}=0 \arrow[d, dash] \\
\alpha_{2,1}=0 \arrow[d, dash] & 0\leq\alpha_{2,1}\leq\alpha_{1,1} \arrow[d, dash] & 0\leq\alpha_{2,1}\leq 1 \arrow[d, dash] \\
\alpha_{2,2}=1 & \alpha_{2,2}=1-\alpha_{2,1} & \alpha_{2,2}=1-\alpha_{2,1}
\end{tikzcd}$$

We have merged some cells for brevity. The important difference to note here is how the cells for $\alpha_{2,1}$ depend on $\alpha_{1,1}$, in order to satisfy IFR.

Importantly, adding the IFR property does not affect simplex-extensibility of system (M).

Lemma 2.38. *If an instance of* (M) *is simplex-extensible, then the same system with the additional requirement of IFR remains simplex-extensible.*

Proof. If an instance of (M) is simplex-extensible, then the same system with the additional requirement of IFR remains simplex-extensible. □

Extensions: Other forms of f^*

Thus far, we have focused on f^* that are linear in the x-type variables. We now generalize our previous result by allowing for functions of the x-type variables, i.e., $f^* = g_0 + \sum_{i=1}^{\eta} f_i g_i$, where f_i is a polynomial of x_i. The main challenge is in keeping track of the projection factors. However, we can generalize it for some polynomials. If, for each i, the polynomial f_i has no non-negative roots, i.e., it is sign-invariant over $x_i > 0$, then all the useful properties about simplex-extensibility and projection-definability carry over.

Corollary 2.39. *Let $f^* = g_0 + \sum_{i=1}^{\eta} f_i g_i$, where each function $\{f_i\}_{i=1}^{\eta}$ is a univariate polynomial of x_i, and $\{g_i\}_{i=0}^{\eta}$ is a (possibly multivariate) polynomial of the α-type variables. If, for each i, f_i has no non-negative roots, then f^* is simplex-extensible if and only if each of $\{g_i\}_{i=0}^{\eta}$ are sign-invariant over the simplex CAD. Furthermore, if* (M) *is simplex-extensible with such an f^*, it is also projection-definable.*

Proof. If a polynomial has no non-negative roots, it is sign-invariant over the non-negative reals. So, for each i, substitute $z_i = f_i$ if f_i is positive or $z_i = -f_i$ if f_i is negative, and treat as a system with z_i, noting that $z_i \geq 0$. The simplex-extensibility follows from Theorem 2.26 and the projection-definability follows from Lemma 2.27. Then, build the CAD and substitute f_i for z_i at the end. □

Some special cases captured by the preceding corollary are when f_i is a monomial with arbitrary exponent, when f_i has all positive coefficients, or when f_i has all negative coefficients, which is implied by Descartes' rule of signs. There are other such f_i as well, which can be easily identified by root-counting algorithms like Sturm sequences [Basu et al., 2006].[4]

Since polynomials with higher order terms require significant computation to compute projection factors, this corollary is valuable because it allows us to easily check simplex-extensibility for systems where f^* has higher order terms in the x-type variables.

Example 2.40. *Let $f^* = \alpha_1(x_1^2 + x_1^3) + \alpha_2 x_2 + \alpha_3 x_3$, with $\alpha_1 + \alpha_2 + \alpha_3 = 1$. If we tried to check for simplex-extensibility by fully computing the projection factors, we would get 13 projection factors when eliminating x_1, 85 when eliminating x_2, and 751 when eliminating x_3. However, with Lemma 2.39 we can easily see that f^* is simplex-extensible.*

The challenge with generalizing to general univariate polynomials (i.e., with non-negative roots) of x-type variables is that, when projecting on a given x-type variable, we will generate multiple projection factors in the other x-type variables, which makes keeping track of projection factors in subsequent steps onerous. This is unlike the case in Theorem 2.26, where only a single projection factor in the other x-type variables is generated, which keeps the subsequent projection factors simple to compute. Indeed, the technical lemmas in Section 2.4.2 exploit this property. Also, in general, projection-definability is not guaranteed either, due to the possibility of multiple roots.

The situation is even more difficult if attempting to generalize to multivariate polynomials of the x-type variables. This form does not provide any special structure, as indeed any f^* can be written as $f_1 g_1 + f_2 g_2 + \cdots$, where the f functions are multivariate polynomials of the x-type variables and the g functions are multivariate polynomials in the α-type variables, simply by taking the individual monomials of f^*.

Fortunately, Theorem 2 provides significant generality already, as it applies to all the relevant sensitivity analyses that we introduced in this chapter. We now return to that context to demonstrate how our results for system (M) apply to our motivating problem.

2.5 Application to Markov Reward Processes

As we discussed at the start of Section 2.4.4, system (M) is a general class that contains the polynomial systems associated with our sensitivity analyses of interest. We formalize this result next.

Theorem 2.41. *Let R_∞ be defined as in (2.3) and $\gamma \in \mathbb{R}$. Then the system $R_\infty \geq \gamma$ with row-stochastic constraints on $\mathbf{P}$ and $\boldsymbol{\pi}$, and non-negativity constraints on $\mathbf{r}$:*

1. *is an instance of system* (M)
2. *is simplex-extensible,*
3. *is projection-definable, and*
4. *has a solution formula that can be constructed via Algorithm 3.*

Proof. Letting $f^* := \boldsymbol{\pi}^\top \operatorname{adj}(\mathbf{I} - \lambda\mathbf{P})\mathbf{r} - \gamma \det(\mathbf{I} - \lambda\mathbf{P})$, $R_\infty \geq \gamma$ is equivalent to $f^* \geq 0$, so the system is an instance of (M). The function f^* can be written in the form $g_0 + \sum_{i=1}^{\eta} x_i g_i$ by setting

[4]Writing out the solution formula for such polynomials may introduce difficulties because of having to represent the roots of high-degree univariate polynomials in the CAD tree. This is an issue that should be handled by the computer algebra system so we do not discuss this further here.

$g_0 = -\gamma \det(\mathbf{I} - \lambda\mathbf{P})$, $x_i = r_i$, and g_i as the ith entry of $\boldsymbol{\pi}^\top \operatorname{adj}(\mathbf{I} - \lambda\mathbf{P})$. Since $\det(\mathbf{I} - \lambda\mathbf{P}) > 0$ (Lemma 2.1) and γ is fixed, g_0 is sign-invariant over the simplex CAD. Similarly, since $\operatorname{adj}(\mathbf{I} - \lambda\mathbf{P})$ is component-wise non-negative (Lemma 2.1), each g_i is non-negative and therefore sign-invariant over the simplex CAD. Thus, the system is simplex-extensible by Theorem 2.26 and projection-definable by Lemma 2.27, and Algorithm 3 can be used to construct its CAD solution formula. □

Several remarks follow about the generalizability of this result.

Remark 2.42 (Finite-horizon reward). The same result holds for the finite-horizon reward with inequality $R_t \geq \gamma$. We set $f^* = R_t - \gamma$, which is of the form $f^* = g_0 + \sum_{i=1}^{\eta} x_i g_i$ with $\{g_i\}_{i=1}^{\eta}$ being the entries in $\sum_{m=0}^{t} \boldsymbol{\pi}^\top \lambda^m \mathbf{P}^m$, per equation (2.1), which are clearly polynomials in $\boldsymbol{\pi}$ and $\mathbf{P}$. Each of these polynomials are formed by the addition of monomials in $\boldsymbol{\pi}$ and $\mathbf{P}$, and so are non-negative. As well, $g_0 = -\gamma$, which is a constant so is sign-invariant. The results follow by invoking Theorem 2.26 and Lemma 2.27.

Remark 2.43 (Other cost-effectiveness quantities). By suitably defining f^*, Theorem 2.41 extends to other sensitivity analyses of interest.

- For comparing the infinite-horizon rewards of two interventions, labeled a and b, the form of the inequality in (2.7) implies that $f^* = \boldsymbol{\pi}_a^\top \operatorname{adj}(\mathbf{I} - \lambda\mathbf{P}_a)\mathbf{r}_a \det(\mathbf{I} - \lambda\mathbf{P}_b) - \boldsymbol{\pi}_b^\top \operatorname{adj}(\mathbf{I} - \lambda\mathbf{P}_b)\mathbf{r}_b \det(\mathbf{I} - \lambda\mathbf{P}_a)$, and the $\{g_i\}_{i=1}^{\eta}$ are the entries in $\boldsymbol{\pi}_a^\top \operatorname{adj}(\mathbf{I} - \lambda\mathbf{P}_a)\det(\mathbf{I} - \lambda\mathbf{P}_b)$ and $-\boldsymbol{\pi}_b^\top \operatorname{adj}(\mathbf{I} - \lambda\mathbf{P}_b)\det(\mathbf{I} - \lambda\mathbf{P}_a)$, which are sign-invariant by similar arguments in the proof above.
- When bounding the NMB, by rearranging equation (2.8) we set $f^* = \boldsymbol{\pi}^\top \operatorname{adj}(\mathbf{I} - \lambda\mathbf{P})(W\mathbf{b} - \mathbf{c}) - \gamma \det(\mathbf{I} - \lambda\mathbf{P})$. Note that the x-type variables are now the elements of $\mathbf{b}$ and $\mathbf{c}$.
- When bounding the ICER of two interventions a and b, we can similarly rearrange (2.9) to obtain a suitable f^*, with the x-type variables being the costs and benefits of each intervention.

Lastly, we note that a benefit of taking the CAD perspective is that we can characterize the exact shape of the boundary defined by any of the cost-effectiveness inequalities of interest as a function of $\boldsymbol{\pi}$ $\mathbf{P}$, and/or $\mathbf{r}$. For example, we can fully characterize the geometry of two-way sensitivity analyses, which we do below.

2.5.1 Geometry of Two-Way Sensitivity Analysis

The formulations of the total reward inequalities allow us to formalize some facts about the geometry of a two-way sensitivity analysis in a general Markov reward process. Suppose we assert $R_\infty \geq \gamma$, which can be reformulated as the inequality $f^* \geq 0$, where f^* is a multilinear polynomial in the Markov chain's parameters. Since f^* is multilinear, if we fix all but two of the parameters, what remains is a polynomial that is either linear (i.e., no variables being multiplied together) or bilinear (i.e., contains a term where both variables are being multiplied together). If it is linear, then the solution to the inequality is a half-plane. If it is bilinear, then the boundary of the solution to the inequality is a hyperbola, and the valid space is one of the sides of this hyperbola. Depending on the shape of the hyperbola and which side the solution lies – which depends on the values of the fixed parameters – the solution is either convex or concave.

Table 2.1: Geometry of two-way sensitivity analysis of the infinite horizon reward of a Markov reward process

Free parameters	Form of f^*	Valid parameter space
Both from $\mathbf{r}$	Bivariate linear	Half-plane
One from $\mathbf{r}$, one from $\boldsymbol{\pi}$ or $\mathbf{P}$	Bilinear or bivariate linear	One side of hyperbola or half-plane
Both from $\boldsymbol{\pi}$ or $\mathbf{P}$		
— On same simplex	Univariate linear	Line segment
— Same column of $\mathbf{P}$	Bivariate linear	Half-plane
— Otherwise	Bilinear or bivariate linear	One side of hyperbola or half-plane

We can discuss these cases more specifically by analyzing which types of parameters are chosen to vary. If both are rewards, then f^* is a linear function, and so the valid parameter space is a half-plane. If one variable is a reward, and another is an entry in $\boldsymbol{\pi}$ or $\mathbf{P}$, then f^* is (possibly) bilinear, due to the two variables being multiplied together. Therefore, the boundary of the valid parameter space is a rational function, and the valid parameter space lies below or above this space. Hence, depending on the boundary and which side of the boundary is valid, we can get a convex region or a concave region. Note that if there is only a single free entry in $\boldsymbol{\pi}$ or $\mathbf{P}$, then it is uniquely identified by the other entries due to the stochastic constraints, so this would reduce to a one-way sensitivity analysis. Hence, for a two-sensitivity analysis to be meaningful, we want this free parameter to be bound by an inequality, e.g., if we have $\pi = [x, 1-x]$.

Lastly, we consider the case where both free parameters are elements of $\boldsymbol{\pi}$ or $\mathbf{P}$. Firstly note that if both free parameters lie on the same simplex, i.e., both are from $\boldsymbol{\pi}$ or both are from the same row of $\mathbf{P}$, then by simple substitution we can rewrite f^* as a linear function in only one of the variables, and solve for its range. Then, on the two-dimensional plane, the set of valid parameter values is in fact only a line segment. On the other hand, if they do not lie on the same simplex, then f^* is bilinear or linear, and the above arguments about the geometry follow. For example, one case where f^* is definitely linear is when the two free parameters are elements of $\mathbf{P}$ that are in the same column. Then, for the entry in the adjugate of $\mathbf{I} - \lambda\mathbf{P}$ corresponding to one of the variables, the other one will be absent. A similar argument, based on the Laplace expansion definition of the determinant, applies to $\det(\mathbf{I} - \lambda\mathbf{P})$ that the two variables will not be multiplied together. We summarize these cases in Table 2.1.

The geometry of the *finite horizon* parameter space is almost the same, as the function f^* is still linear in the entries of $\mathbf{r}$ and linear in the entries of $\boldsymbol{\pi}$. However, due to summing exponents of $\mathbf{P}$, the entries of $\mathbf{P}$ appear with exponents (i.e., with degrees higher than 1, unless the reward is only computed for a single period). So, if an entry from $\mathbf{P}$ is chosen, we possibly obtain a polynomial with degree greater than one – not necessarily a bilinear function as in the infinite horizon case. In this case, it is difficult to solve exactly for one variable in terms of another, as to do so may require an arduous expression with radicals. And indeed, it may be impossible if we sum the rewards of the Markov process for 5 periods or more, due to the Abel-Ruffini Theorem [Ruffini, 1799, Abel, 1824]. However, the boundary is still always either convex or concave, and depending on the shape of the curve and which side of the curve we are on, the valid parameter space may be non-convex.

2.6 Software: `markovag`

We developed a Python package, `markovag`, available at `https://github.com/mmaaz-git/markovag`, that allows practitioners to use CAD for sensitivity analyses. The package uses SymPy [Meurer et al., 2017], which is a computer algebra system in Python, to perform symbolic algebraic manipulation. There are two modules.

The first module, `markovag.markov`, can perform computations with symbolic Markov chains, i.e., where some of the parameters are variables. It contains functions to parse symbolic matrices from input files. One can then perform matrix computations on them. The module comes with functions to symbolically calculate the finite or infinite horizon discounted rewards, as well as the usual health economic metrics of interest like NMB and ICER. In other words, these functions generate the polynomials in equations (2.3) – (2.9) and their finite horizon analogues. It also provides plotting functions to visualize two- and three-way sensitivity analyses.

The second module, `markovag.cad`, constructs the CAD. It takes polynomials generated by the first module, appends the necessary stochastic constraints, and builds the CAD tree in accordance with Algorithm 3. This development is significant from the larger perspective of computer algebra software, as SymPy currently does not have a CAD solver. In fact, there are only two available implementations of CAD: Mathematica [Wolfram Research, 2020], which is proprietary, and QEPCAD [Brown, 2003], which is open-source. However, both programs require specific syntax and do not come with specialized methods for cost-effectiveness analysis. Since `markovag` is built on Python, it is freely available and leverages syntax that is widely used.

We also note that the two state-of-the-art satisfiability modulo theories solvers, Z3 [De Moura and Bjørner, 2008, Jovanović and De Moura, 2013] and CVC5 [Barbosa et al., 2022, Kremer et al., 2022] both implement CAD. However, they are satisfiability solvers and so only provide a satisfying point, or indicate that none exist. They do not implement the solution formula construction step. In other words, they only implement Algorithm 1 but not Algorithm 2.

Although `markovag` only works for instances of system (M) specifically resulting from Markov reward processes, our implementation has many of the fundamentals for constructing a general CAD, including sample point computation and building the tree data structure. Building this out into a full CAD solver is left to future work.

We now demonstrate a synthetic case study using `markovag`. Later, in Chapter 4, we will show a more fulsome case study.

2.6.1 Synthetic Case Study

Consider a three-state Markov chain, where the states represent (1) "healthy", (2) "sick", and (3) "dead". We set the reward of "healthy" to r_1, the reward of "sick" to r_2 and the reward of "dead" to 0. We consider death to be an absorbing state and we compute the infinite horizon expected total reward per equation (2.10). The total reward is a function of the following five parameters: $p_{11}, p_{12}, p_{21}, p_{22}, r_1, r_2$. Observe that they are not a function of p_{13} nor p_{23}, i.e., the mortalities of each state, which is implied by equation (2.10).

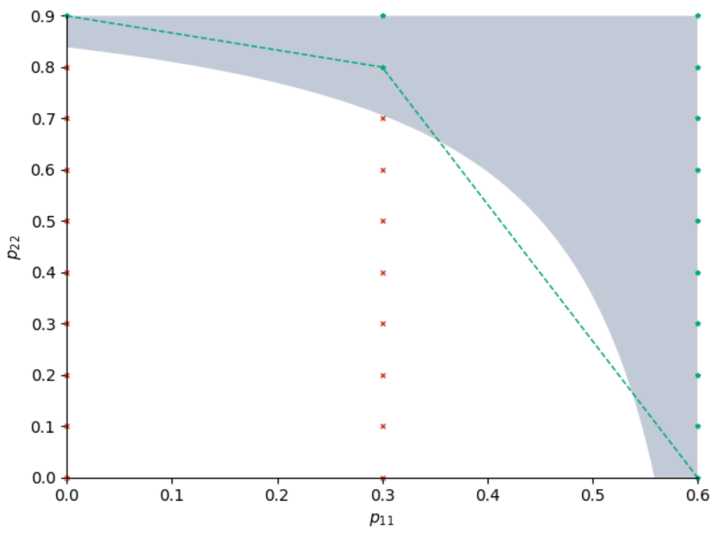


Figure 2.1: Valid parameter space of a $n = 3$ state Markov chain with states (1) healthy, (2) sick, (3) dead, as discussed in Section 2.6.1, with the parameters $p_{12} = 0.4, p_{21} = 0.1, r_1 = 1, r_2 = 0.5$. We assert that $R_\infty \geq 3$. Green and red points correspond to the grid search, where green indicates a valid point and red an invalid one. The green lines form the convex hull of the green points.

Two-way analysis

First, we conduct a two-way sensitivity analysis of the infinite horizon reward. We fix $p_{12} = 0.4, p_{21} = 0.1, r_1 = 1, r_2 = 0.5$, and let p_{11} and p_{22} vary. Given the fixed parameters, we have the implied bounds $p_{11} \in [0, 0.6]$ and $p_{2,2} \in [0, 0.9]$. We wish to find the values of p_{11} and p_{22} that guarantee $R_\infty \geq 3$. Using `markovag.markov`, we symbolically compute the expected total reward, form the inequality bounding it, and plot the resulting valid parameter space (Figure 2.1).

Notice that this parameter space is *not convex*. Without CAD, the traditional approach of using a fixed parameter grid would require shrinking the grid size to get a good approximation of the boundary. However, it would not be known a priori how small the grid needs to be to achieve a desired error. Our exact approach circumvents this issue.

In Figure 2.1, we overlay a grid of lattice points, where green points satisfy the inequality and red points do not. The naive approach of identifying neighboring grid points that satisfy the inequality (i.e., cost-effective) may lead to an incorrect conclusion that all convex combinations of those parameter values also satisfy the inequality.

Another result is that at $p_{11} = 0.3$, we can tell from the grid that $p_{22} = 0.8$ the inequality is satisfied by at $p_{22} = 0.7$ it is not. So the traditional analysis only knows that the cost-effectiveness boundary is crossed somewhere in the range $[0.7, 0.8]$, whereas the CAD analysis clearly shows that the boundary is very close to 0.7.

Multi-way analysis via CAD

To illustrate the power of the CAD approach, we consider an eight-way sensitivity analysis. We allow all of $p_{11}, p_{12}, p_{13}, p_{21}, p_{22}, p_{23}, r_1, r_2$ to vary. The difficulty in searching and visualizing this high-dimensional grid means that an eight-way analysis is most likely never done in practice. We once again assert that $R_\infty \geq 3$. Using the `markovag` package we construct the CAD tree and restrict our attention to the full-dimensional cells (cells which are intervals) for the probabilities. We explore the effect of the probabilities on the permissible rewards, as well as the interaction between the two rewards r_1 and r_2. For example, we find that if

$$r_1 \geq \frac{3p_{11}p_{22} - 3p_{11} - 3p_{12}p_{21} - 3p_{22} + 3}{1 - p_{22}}$$

then $r_2 \geq 0$. This result is useful, for example, if the reward for being in the healthy state is well-known and accepted but the reward for being in the sick state is highly uncertain and patient dependent. A policymaker only needs to know that if a patient receives a sufficiently high reward for being in the healthy state, then any reward value in the sick state would still lead to a cost-effective result. Furthermore, if the reward in state 1 can be written as benefit minus cost, a lower bound on r_1 implies an upper bound on cost, such that if the policymaker can bring the cost of a hypothetical therapeutic below the bound, then the intervention would be considered cost-effective even without knowing the exact reward for being in state 2.

Another benefit of our approach is that it immediately elucidates the impact of adding or removing restrictions on the parameter values. For example, the analysis up to now did not include the IFR condition. Without this condition, the CAD analysis shows that, e.g., $p_{11} = 0.2$, $p_{12} = 0.5$, $p_{21} = 0.3$, $p_{22} = 0.5$, $r_1 \geq 1.5$, $r_2 \geq 0$ results in $R_\infty \geq 3$, suggesting that this combination of parameters is associated with a cost-effective intervention. However, with the IFR condition imposed, the CAD tree would no longer contain this set of parameter values due to the additional restrictions imposed on p_{21} and p_{22}. Lowering p_{21} to satisfy IFR, e.g., $p_{21} = 0.1$, would result in $r_1 \geq 2.15$, $r_2 \geq 0$. This result implies that a higher reward is needed in the healthy state to retain cost-effectiveness.

2.7 Conclusion

In this chapter, we studied the problem of performing sensitivity analysis on metrics that are derived from Markov reward processes. We were motivated specifically by cost-effectiveness analyses in healthcare. We showed that every such analysis has an equivalent semialgebraic representation. Our framework can encompass other common extensions in the healthcare literature, like the IFR assumption. We then showed that these systems have a special structure which allows us to construct their CADs in a simpler way. We call this property simplex-extensible, and it is more general than just the polynomials arising from our study of Markov reward processes. Lastly, we developed software which allows practitioners to use our approach, and we demonstrated that it finds regions of cost-effectiveness that are missed by the classical grid search method.

Chapter 3

Formal Verification of Markov Processes with Learned Parameters

3.1 Introduction

In this chapter, we study systems where machine learning (ML) models are integrated into Markov models. As ML becomes increasingly integrated into real-world systems, Markov models are evolving to incorporate heterogeneous, data-driven parameters. For instance, ML models can estimate failure rates from sensor data or patient-specific transition probabilities based on clinical features [Mertens et al., 2022]. This gives rise to a new class of Markov processes where parameters are not fixed but are instead *learned functions*.

Despite this growing trend, there is limited work on rigorously analyzing such systems. In healthcare, most studies rely on Monte Carlo simulation [Krijkamp et al., 2018], which supports subgroup analyses (e.g., patients over 60) but cannot provide formal guarantees – an important limitation in high-stakes domains like medicine or safety-critical infrastructure.

In this chapter, we present a new framework for the formal verification of Markov processes with ML-based parameters. Our key insight is that properties of such systems can be encoded as *bilinear programs*, allowing us to obtain exact results with formal guarantees. While our primary motivation is healthcare, the approach generalizes to any application where learned models feed into Markov processes. This enables us to rigorously answer questions such as: Given a bound on the input, what is the worst-case probability of reaching a failure state? Is the failure rate for machines with certain properties guaranteed to remain below 0.01%? For a clinical subgroup, is the expected treatment cost within a government threshold?

To summarize, our main contributions are:

1. We introduce a general framework for formally verifying properties of Markov processes whose parameters are given by ML models.
2. We show that for a broad class of models, including linear models, tree ensembles, and ReLU-based neural networks, verification problems can be expressed as (mixed-integer) bilinear programs.

3. We develop a novel decomposition and bound propagation method that significantly accelerates global solution times, outperforming existing solvers by orders of magnitude.

3.2 Related Work

Formal Verification of Markov Processes Probabilistic model checking verifies properties of systems modeled by Markov chains [Hansson and Jonsson, 1994, Baier and Katoen, 2008], using tools like PRISM and others [Kwiatkowska et al., 2002, Katoen et al., 2005, Hermanns et al., 2000, Sen et al., 2005, Younes, 2005, Lassaigne and Peyronnet, 2002]. More recent techniques allow parameters to be intervals [Sen et al., 2006, Delahaye et al., 2015, Petrucci and van De Pol, 2018], rational functions [Chen et al., 2013, Junges, 2020, Junges et al., 2024], or distributions [Badings et al., 2023]. We extend this further by allowing parameters defined by ML models, and release our own tool, `markovml`.

Formal Verification of ML Models Formal verification of ML models aims to prove properties such as robustness or output bounds; a well-known benchmark is ACAS Xu [Owen et al., 2019]. ReLU neural networks can be encoded as satisfiability modulo theories (SMT) or mixed-integer linear programming (MILP) problems, enabling verification via tools like Reluplex [Katz et al., 2017], a modified simplex method that spurred substantial follow-up work [Tjeng et al., 2017, Cheng et al., 2017, Anderson et al., 2020, Tjandraatmadja et al., 2020, Kronqvist et al., 2021, Anh-Nguyen and Huchette, 2022]. State-of-the-art methods such as α, β-CROWN combine bound propagation, branch-and-bound, and GPU acceleration [Wang et al., 2021, Zhang et al., 2022b,a, Kotha et al., 2023]. We leverage these formulations to embed ML models directly into our optimization framework. Formal verification of ML is also related to the field of adversarial ML, which attempts to find adversarial examples, or to protect against them; see, e.g., Carlini and Wagner [2017].

Bilinear Programming Bilinear programs are NP-hard non-convex quadratic problems in which the objective or constraints contain bilinear terms [Horst and Pardalos, 2013]. Global optima can be computed via branch-and-bound [Horst and Pardalos, 2013], aided by convex relaxations like McCormick's envelope [McCormick, 1976], and are supported by modern solvers such as Gurobi [Gurobi Optimization, LLC, 2024]. There are a number of results on relaxations and formulations for bilinear programs: e.g., if the variables live in two separate polyhedra, the optimum is a vertex from each polyhedra, so the problem can be rewritten as a MILP [Horst and Pardalos, 2013], or, if the bilinear program has a bilinear structure, then there is a second-order cone relaxation. However, our problem does not satisfy these, necessitating a new approach. Moreover, we find that Gurobi's default approach performs poorly on our class of problems. In contrast, our method accelerates solution time by orders of magnitude while still leveraging Gurobi's global optimality guarantees.

Related Software There exist several tools for *probabilistic model checking*, which verifies properties of systems modeled by Markov chains [Hansson and Jonsson, 1994, Baier and Katoen, 2008], e.g., PRISM and others [Kwiatkowska et al., 2002, Katoen et al., 2005, Hermanns et al., 2000, Sen et al., 2005, Younes, 2005, Lassaigne and Peyronnet, 2002]. Concurrently, there also exist several tools for verifying machine learning models, which take a pretrained model and obtain guaranteed bounds on outputs, e.g., α, β-CROWN [Wang et al., 2021, Zhang et al., 2022b,a, Kotha et al., 2023], VeriNet [Henriksen and Lomuscio, 2020, 2021], and Marabou [Katz et al., 2019, Wu et al., 2020]. In the healthcare domain, simulation software like TreeAge [TreeAge Software, 2021] allow for heterogeneous parameters of the Markov chain drawn from distributions, but currently only supports Monte Carlo simulation. Our software is thus the first tool that allows the formal verification of integrated Markov models and ML models, and, by formulating the problem as an optimization problem and solving it to global optimality, it provides robust guarantees that simulation cannot.

3.3 Problem Formulation

We have, as in Section 1.5, a Markov chain with n states, with a row-stochastic transition matrix $\mathbf{P} \in \mathbb{R}^{n \times n}$, and a stochastic initial distribution vector $\boldsymbol{\pi} \in \mathbb{R}^n$. We may also have a reward vector $\mathbf{r} \in \mathbb{R}^n$ and form a Markov reward process.

In this chapter, we will study how to verify three commonly computed properties, defined in Chapter 1.5: reachability, expected hitting time, and total infinite-horizon discounted reward. These three quantities enable rich analysis of Markov processes and the systems they model. Reachability can be used to verify the probability of failure in a system. Expected hitting time can be used to compute the expected time to failure, or life expectancy of a person. Total reward can be used to compute the resource consumption of a system or total cost of a drug.

We wish to *verify* these properties, namely finding their maximum or minimum. If the Markov process' parameters are fixed, as is common in model checking, these quantities can be computed exactly by solving a linear system. However, in our case, the parameters are given by ML models, and so we will formulate an optimization problem to derive bounds on these quantities. As the three quantities have similar formulations (see their formulas in Chapter 1.5), we will proceed in the rest of the paper by studying the total reward. It will be easy to modify results for the other two quantities.

3.3.1 Embedding ML Models

We now consider $\boldsymbol{\pi}, \mathbf{P}, \mathbf{r}$ as functions of a *feature vector* $\mathbf{x} \in \mathbb{R}^m$, where m is the number of features. We encode the relationship from $\mathbf{x}$ to $\boldsymbol{\pi}, \mathbf{P}, \mathbf{r}$ via a set of functions, $f_1, f_2, \ldots, f_{k_f}$, where k_f is the number of functions. Each function $f_i : \mathbb{R}^m \to \mathbb{R}^{\ell_i}$ for $i = 1, 2, \ldots, k_f$ takes a feature vector and outputs a vector $\boldsymbol{\theta}_{\mathbf{i}}$ (e.g., a classifier may output class probabilities for three classes). We concatenate these vectors to form the output vector $\boldsymbol{\theta} \in \mathbb{R}^{\ell}$, where $\ell = \sum_{i=1}^{k_f} \ell_i$.

Our key assumption on the functions will be that they are *mixed-integer linear representable* (MILP-representable), meaning that the relationship between the inputs and outputs can be expressed using linear constraints and binary variables. This is a broad class of functions that includes, e.g., piecewise linear functions (including simple "if-then" rules), linear and logistic regression, tree-based models, and neural networks with ReLU activations.

It is important to note that in the case of logistic regression and neural network classifiers, there is a non-linearity introduced by the softmax or logistic function. Typically in the formal verification literature, it suffices to check the logits, not the actual probabilities. However, in our problem, we need the probabilities as they are inputs into our Markov process. Practically, we handle this issue in this work and in our software using Gurobi's built-in *spatial branch-and-bound techniques* for nonlinear functions – essentially dynamic piecewise linear approximations. Note also that this framework also includes more exotic types of models like generalized additive models where the underlying functions are MILP-representable, e.g., sums of trees, as in explainable boosting machines [Lou et al., 2013].

Next, the output vector $\boldsymbol{\theta}$ is linked to $\boldsymbol{\pi}, \mathbf{P}$ and $\mathbf{r}$ through affine equalities: $\boldsymbol{\pi} = \mathbf{A}_{\boldsymbol{\pi}}\boldsymbol{\theta} + \mathbf{b}_{\boldsymbol{\pi}}$, $\text{vec}(\mathbf{P}) = \mathbf{A}_{\mathbf{P}}\boldsymbol{\theta} + \mathbf{b}_{\mathbf{P}}$, and $\mathbf{r} = \mathbf{A}_{\mathbf{r}}\boldsymbol{\theta} + \mathbf{b}_{\mathbf{r}}$, for fixed $\mathbf{A}_{\boldsymbol{\pi}} \in \mathbb{R}^{n \times \ell}, \mathbf{A}_{\mathbf{P}} \in \mathbb{R}^{n^2 \times \ell}, \mathbf{A}_{\mathbf{r}} \in \mathbb{R}^{n \times \ell}$ and $\mathbf{b}_{\boldsymbol{\pi}} \in \mathbb{R}^n, \mathbf{b}_{\mathbf{P}} \in \mathbb{R}^{n^2}, \mathbf{b}_{\mathbf{r}} \in \mathbb{R}^n$. Note the sizes of the matrices mean that every element of $\boldsymbol{\pi}, \mathbf{P}, \mathbf{r}$ must be bound with an affine equality to $\boldsymbol{\theta}$. In other words, if we fix $\boldsymbol{\theta}$, we can compute $\boldsymbol{\pi}, \mathbf{P}, \mathbf{r}$ exactly. The vec operator vectorizes a matrix, i.e., by concatenating rows into one column vector, so that $\text{vec}(\mathbf{P}) \in \mathbb{R}^{n^2}$.

The affine equalities allow for common formulations like one of the parameters equaling an element of $\boldsymbol{\theta}$, e.g., $\pi_1 = \theta_1$. Another common situation is that we have a function that outputs the probability of transitioning to a specific state (say, a death state in a healthcare model), and the remaining probability mass is, e.g., distributed uniformly over the remaining states: $\pi_n = \theta_1$ and $\pi_i = (1 - \sum_{j \neq i} \theta_j)/(n-1)$ for $i \neq n$. Of course, the affine equalities also allow for parameters to be fixed as constants.

Next, we may have linear inequalities on the parameters: $\mathbf{C}_{\boldsymbol{\pi}}\boldsymbol{\pi} \leq \mathbf{d}_{\boldsymbol{\pi}}$, $\mathbf{C}_{\mathbf{P}}\,\text{vec}(\mathbf{P}) \leq \mathbf{d}_{\mathbf{P}}$, and $\mathbf{C}_{\mathbf{r}}\mathbf{r} \leq \mathbf{d}_{\mathbf{r}}$, for fixed $\mathbf{C}_{\boldsymbol{\pi}} \in \mathbb{R}^{k_{\boldsymbol{\pi}} \times n}, \mathbf{C}_{\mathbf{P}} \in \mathbb{R}^{k_{\mathbf{P}} \times n^2}, \mathbf{C}_{\mathbf{r}} \in \mathbb{R}^{k_{\mathbf{r}} \times n}$ and $\mathbf{d}_{\boldsymbol{\pi}} \in \mathbb{R}^{k_{\boldsymbol{\pi}}}, \mathbf{d}_{\mathbf{P}} \in \mathbb{R}^{k_{\mathbf{P}}}, \mathbf{d}_{\mathbf{r}} \in \mathbb{R}^{k_{\mathbf{r}}}$, where $k_{\boldsymbol{\pi}}, k_{\mathbf{P}}, k_{\mathbf{r}}$ are the number of linear inequalities on $\boldsymbol{\pi}$, $\mathbf{P}$, and $\mathbf{r}$, respectively.

Linear inequalities are an appropriately general assumption as they arise naturally from fixing a monotonic ordering on the rewards or the probabilities. In reliability engineering and healthcare, a common assumption is that of *increasing failure rate*, which implies a certain stochastic ordering on the transition probabilities [Barlow and Proschan, 1996, Alagoz et al., 2007]. Effectively, this introduces a series of linear inequalities on the transition probabilities. See Figure 3.1 for a diagram of the full pipeline.

Lastly, we assume that the feature vector $\mathbf{x}$ lies in a set $\mathcal{X} \subseteq \mathbb{R}^m$. We assume that the set $\mathcal{X}$ is *mixed-integer-linear representable* (MILP-representable), i.e., it can be written as a finite union of polyhedra [Jeroslow and Lowe, 1984]. This is satisfied by a wide range of sets, including box constraints, polytopes, and discrete sets. In practice, this may look like, e.g., that we restrict $\mathbf{x}$ to represent the set of patients that are either under 18 or over 65, as this is a union of two intervals.

With these assumptions, we can now formulate the following optimization problem for the total

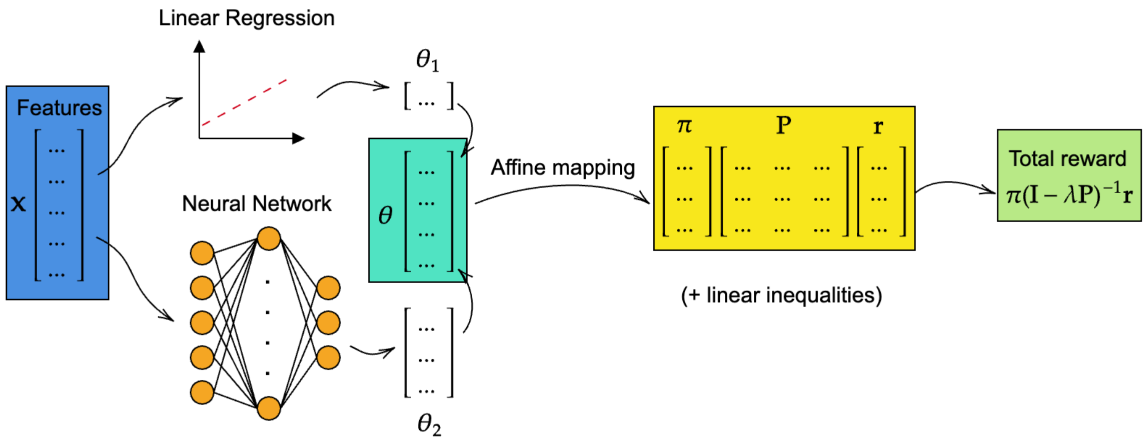


Figure 3.1: Example of our pipeline. A feature vector $\mathbf{x}$ is passed through different functions, here a linear regression and a neural network, to obtain the output vector $\boldsymbol{\theta}$, which then determines the parameters of the Markov process through affine equalities.

infinite-horizon discounted reward:

$$
\begin{aligned}
&\min/\max_{\boldsymbol{\pi},\mathbf{P},\mathbf{r},\mathbf{v},\mathbf{x},\boldsymbol{\theta}} \boldsymbol{\pi}^\top \mathbf{v} \quad \text{s.t.}\\
&\quad \mathbf{v} = \lambda \mathbf{P}\mathbf{v} + \mathbf{r},\\
&\quad \boldsymbol{\theta} = \left[\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \cdots, \boldsymbol{\theta}_{k_f}\right]^\top,\\
&\quad \boldsymbol{\theta}_i = f_i(\mathbf{x}) \quad \forall i \in [k_f],\\
&\quad \boldsymbol{\pi} = \mathbf{A}_{\boldsymbol{\pi}}\boldsymbol{\theta} + \mathbf{b}_{\boldsymbol{\pi}},\\
&\quad \operatorname{vec}(\mathbf{P}) = \mathbf{A}_{\mathbf{P}}\boldsymbol{\theta} + \mathbf{b}_{\mathbf{P}},\\
&\quad \mathbf{r} = \mathbf{A}_{\mathbf{r}}\boldsymbol{\theta} + \mathbf{b}_{\mathbf{r}},\\
&\quad \mathbf{C}_{\boldsymbol{\pi}}\boldsymbol{\pi} \le \mathbf{d}_{\boldsymbol{\pi}},\\
&\quad \mathbf{C}_{\mathbf{P}} \operatorname{vec}(\mathbf{P}) \le \mathbf{d}_{\mathbf{P}},\\
&\quad \mathbf{C}_{\mathbf{r}}\mathbf{r} \le \mathbf{d}_{\mathbf{r}},\\
&\quad \sum_{j=1}^{n} P_{ij} = 1 \quad \forall i \in [n],\\
&\quad \sum_{i=1}^{n} \pi_i = 1,\\
&\quad 0 \le P_{ij} \le 1 \quad \forall i,j \in [n]\times[n],\\
&\quad 0 \le \pi_i \le 1 \quad \forall i \in [n],\\
&\quad \mathbf{x} \in \mathcal{X}.
\end{aligned}
$$

Above, $\mathbf{v}$ is a new vector we have introduced, which in Markov process theory would be referred to as the *value function*, and often arises in reinforcement learning, which equals $\mathbf{v} = (\mathbf{I} - \lambda\mathbf{P})^{-1}\mathbf{r}$. We can easily convert this into a feasibility problem, to find a feasible $\mathbf{x}$, by replacing the objective function with an inequality, $W_{\min} \le \boldsymbol{\pi}^\top \mathbf{v} \le W_{\max}$. With slight modifications, we can also formulate reachability and hitting time: the full formulations are in Section 3.5.

3.4 Solving the Optimization Problem

Under the assumptions that the functions $f_1, f_2, \ldots, f_{k_f}$ are MILP-representable and $\mathcal{X}$ is MILP-representable, the optimization problem above is a *mixed-integer bilinear program*, as we have a bilinear objective and a bilinear constraint. We will leverage the observation that strong bounds on variables are crucial for solving bilinear programs, and will derive such bounds by decomposing our problem and using various results from Markov theory.

Our strategy is as follows: obtain bounds on the elements of $\boldsymbol{\theta}$ by solving a series of smaller MILPs, propagate these bounds to the parameters via the affine equalities, obtain bounds on $\mathbf{v}$ using linear algebra arguments, and then tighten the $\mathbf{v}$ bounds using interval matrix analysis.

Bounds on $\boldsymbol{\theta}$ For each θ_i, we minimize and maximize it by solving two smaller MILPs, over the set $\mathcal{X}$. Namely, for each $i \in [k_f]$, for each $j \in [\ell_i]$, solve for $\min/\max_{\mathbf{x}\in\mathcal{X}} \theta_{i,j}$ s.t. $\boldsymbol{\theta}_\mathbf{i} = [\theta_{i,1}, \ldots, \theta_{i,\ell_i}] = f_i(\mathbf{x})$, which are MILPs as $\mathcal{X}$ and $f_1, \ldots f_{k_f}$ are MILP-representable.

Bounds on $\boldsymbol{\pi}, \mathbf{P}, \mathbf{r}$ We now propagate the bounds on $\boldsymbol{\theta}$ to the parameters $\boldsymbol{\pi}, \mathbf{P}, \mathbf{r}$ via the affine equalities.

Lemma 3.1. *Let $\boldsymbol{\theta} \in \mathbb{R}^\ell$ be a vector where each component satisfies the bounds $\theta_i^{\min} \leq \theta_i \leq \theta_i^{\max}$ for all $i = 1, 2, \ldots, \ell$. Consider an affine transformation defined by $\mathbf{y} = \mathbf{A}\boldsymbol{\theta} + \mathbf{b}$, where $\mathbf{y} \in \mathbb{R}^k$, $\mathbf{A} \in \mathbb{R}^{k\times\ell}$, and $\mathbf{b} \in \mathbb{R}^k$. Then, each component y_i of $\mathbf{y}$ satisfies $b_i + \sum_{j=1}^{\ell} A_{ij} \cdot \left(\theta_j^{\min}\mathbb{1}_{A_{ij}\geq 0} + \theta_j^{\max}\mathbb{1}_{A_{ij}<0}\right) \leq y_i \leq b_i + \sum_{j=1}^{\ell} A_{ij} \cdot \left(\theta_j^{\max}\mathbb{1}_{A_{ij}\geq 0} + \theta_j^{\min}\mathbb{1}_{A_{ij}<0}\right)$.*

Proof. For a given component y_i, we can write the transformation as:

$$y_i = \sum_{j=1}^{\ell} A_{ij}\theta_j + b_i$$

for all $i = 1, 2, \ldots, k$.

Each term in the summation depends linearly on θ_j. If $A_{ij} \geq 0$, then $A_{ij}\theta_j$ is maximized at $\theta_j^{\max}$ and minimized at $\theta_j^{\min}$. Vice versa, if $A_{ij} < 0$, then $A_{ij}\theta_j$ is maximized at $\theta_j^{\min}$ and minimized at $\theta_j^{\max}$.

Applying this to all terms in the summation, we get the required bounds. □

Bounds on $\mathbf{v}$ Next, we derive bounds on $\mathbf{v}$ using well-established facts from Markov theory. First, as $\mathbf{v} = (\mathbf{I} - \lambda\mathbf{P})^{-1}\mathbf{r}$, we analyze bounds on $(\mathbf{I} - \lambda\mathbf{P})^{-1}$.

Lemma 3.2. *Let $\mathbf{P}$ be row-stochastic and $\lambda \in (0, 1)$. Then, each element in $(\mathbf{I} - \lambda\mathbf{P})^{-1}$ is in the interval $[0, 1/(1 - \lambda)]$ and the row sums are all equal to $1/(1 - \lambda)$.*

Proof. Note that the matrix $(\mathbf{I} - \lambda\mathbf{P})$ is a (non-singular) M-matrix. By Theorem 2.3 in Chapter 6 of [Berman and Plemmons, 1994], each element of the matrix is non-negative, so is ≥ 0.

Next, consider the row sum of $(\mathbf{I} - \lambda\mathbf{P})^{-1}$ for a row i: $\sum_j (\mathbf{I} - \lambda\mathbf{P})^{-1}_{ij}$, and then expand into the Neumann series: $\sum_j \sum_{k=0}^{\infty} \lambda^k (\mathbf{P}^k)_{ij}$. We can exchange the order of summation as all terms are non-negative, and then we have our sum is $\sum_{k=0}^{\infty} \lambda^k \sum_j (\mathbf{P}^k)_{ij}$. Every positive integer exponent of a row-stochastic matrix is also row-stochastic, so $\sum_j (\mathbf{P}^k)_{ij} = 1$, for all $k \in \mathbb{N}$. Therefore, this simplifies to the geometric series $\sum_{k=0}^{\infty} \lambda^k$, which equals $\frac{1}{1-\lambda}$. As each row sum is $\frac{1}{1-\lambda}$, and each element is non-negative, each element must also be at most $\frac{1}{1-\lambda}$. □

Now we can propagate the above bounds to $\mathbf{v}$.

Lemma 3.3. *Let $r_i^{\min} \leq r_i \leq r_i^{\max}$. Then, each element in the vector $\mathbf{v}$ satisfies $\gamma \min_j r_j^{\min} \leq v_i \leq \gamma \max_j r_j^{\max}$, where $\gamma = \frac{1}{1-\lambda}$.*

Proof. Recall that $\mathbf{v} = (\mathbf{I} - \lambda\mathbf{P})^{-1}\mathbf{r}$, so $v_i = \sum_{j=1}^{n} (\mathbf{I} - \lambda\mathbf{P})^{-1}_{ij} r_j$. Thus, we can see v_i is a positive linear combination of the elements of $\mathbf{r}$, where the nonnegative weights are from $(\mathbf{I} - \lambda\mathbf{P})^{-1}$ and sum to $1/(1-\lambda)$. To maximize this combination, we assign all the weight to the maximum upper bound of $\mathbf{r}$, and to minimize it, we assign all the weight to the minimum lower bound of $\mathbf{r}$. Hence, we have the required bounds. □

Lemma 3.3 provides valid bounds on $\mathbf{v}$. However, these bounds are not tight, as Lemma 3.2 applies to *any* row-stochastic matrix, and does not take into account the specific element-wise bounds on $\mathbf{P}$ that we derived from Lemma 3.1. In order to derive tighter bounds on $\mathbf{v}$, we need to incorporate the specific element-wise bounds on $\mathbf{P}$. This is difficult because of the matrix inverse. In the following section, we show how to solve this problem with interval matrix analysis and hence derive tighter bounds on $\mathbf{v}$.

3.4.1 Tightening Bounds on $\mathbf{v}$

Interval matrix analysis studies a generalization of matrices (or vectors), where each element is an *interval* instead of a real number (see [Neumaier, 1984] for an overview). Equivalently, it can be viewed as studying the set of all matrices that lie within given intervals. There are easy generalizations of arithmetic of real numbers to intervals, which can be used to generalize matrix arithmetic to interval matrices. However, inverting an interval matrix is difficult due to the inherent nonlinearity of the inverse operation. Consider the equivalent problem of solving a linear system with interval matrices. The solution set is not rectangular, so cannot be described as an interval matrix. Hence, we wish to find the interval matrix of smallest radius that contains the solution set, known as the *hull* [Neumaier, 1984].

Due to the interval extensions of the basic arithmetic operations, it is possible to generalize the Gauss-Seidel method, a common iterative procedures for solving linear systems, to interval matrices. Beginning with an initial enclosure of the solution set, it iteratively refines the intervals and returns a smaller enclosure of the solution set. We provide a full description of the Gauss-Seidel method below. The interval Gauss-Seidel method is optimal in the sense that it provides the smallest enclosure of the solution set out of a broad class of algorithms [Neumaier, 1984].

Gauss-Seidel method for interval matrices

Our exposition here follows section 5.7 of Horáček [2019]. Suppose we have a linear system $\mathbf{Ay} = \mathbf{b}$, where $\mathbf{A}$ is an interval matrix and $\mathbf{b}$ is an interval vector. The set of $\mathbf{y}$ that satisfies this system is referred to as the solution set. It is in general complicated, so we hope to find a rectangular enclosure of the solution set. The smallest such enclosure is referred to as the hull of the solution set.

Suppose we have an initial enclosure of the solution set, $\mathbf{y}^{(0)}$. In general this takes some effort to obtain, but we obtained this for our specific problems. Then, the interval Gauss-Seidel method proceeds as follows:

1. For each variable x_i, compute a new enclosure of the solution set as:

$$z_i^{(k+1)} = \frac{1}{A_{ii}} \left(b_i - \sum_{j<i} A_{ij} y_j^{(k+1)} - \sum_{j>i} A_{ij} y_j^{(k)} \right)$$

2. Update the enclosure of the solution set: $\mathbf{y}^{(k+1)} = \mathbf{y}^{(k)} \cap \mathbf{z}^{(k+1)}$.
3. Repeat steps 1 and 2 until stopping criterion is met (either a maximum number of iterations or the difference in subsequent enclosures is less than some tolerance).

All operations above are interval arithmetic operations. For real intervals, we have: $[a,b]+[c,d] = [a+c, b+d]$, $[a,b]-[c,d] = [a-d, b-c]$, $[a,b]\cdot[c,d] = [\min(ac,ad,bc,bd), \max(ac,ad,bc,bd)]$. Division is defined as multiplication by the reciprocal, i.e.,

$$[a,b]/[c,d] = [a,b]\cdot[1/d, 1/c] = [\min(a/c, a/d, b/c, b/d), \max(a/c, a/d, b/c, b/d)],$$

if $0 \notin [c,d]$.

Note that the zero division issue is not a concern for us, as the diagonal elements of $\mathbf{I} - \lambda\mathbf{P}$, in the total reward setting, have lower bounds always strictly positive, due to the discount factor $\lambda \in (0,1)$. In the reachability and hitting time settings, the diagonal elements of $\mathbf{I} - \mathbf{Q}$ have lower bounds always strictly positive, due to the strict substochastic property of $\mathbf{Q}$. In practice, we use a small numerical offset to handle this.

Applying Gauss-Seidel

Hence, we can apply the Gauss-Seidel method to tighten the bounds on $\mathbf{v}$, as $(\mathbf{I} - \lambda\mathbf{P})\mathbf{v} = \mathbf{r}$ is a linear system. We use the bounds on $\mathbf{P}$ and $\mathbf{r}$ from Lemma 3.1 to form the interval matrices, and for our initial guess, we can use the bounds derived from Lemma 3.3.

Still, Gauss-Seidel is not guaranteed to obtain the hull itself. A well-known sufficient condition for Gauss-Seidel to obtain the hull is that the matrix in the linear system is an *interval M-matrix*, a generalization of the notion of an M-matrix [Berman and Plemmons, 1994] to interval matrices. Suppose a matrix $\mathbf{M}$ is element-wise bounded by $\mathbf{M}^{\mathbf{max}}$ and $\mathbf{M}^{\mathbf{min}}$. It is an *interval M-matrix* if and only if $\mathbf{M}^{\mathbf{min}}$ is a (non-singular) real M-matrix in the usual sense and $\mathbf{M}^{\mathbf{max}}$ has no positive off-diagonal elements [Neumaier, 1984]. Hence we can prove a necessary and sufficient condition for $\mathbf{I} - \lambda\mathbf{P}$ to be an interval M-matrix.

Theorem 3.4. *Let* $\mathbf{P}$ *be a row-stochastic matrix bounded element-wise by* $\mathbf{P}^{\mathbf{min}} \leq \mathbf{P} \leq \mathbf{P}^{\mathbf{max}}$. *Then,* $\mathbf{I} - \lambda\mathbf{P}$ *is an interval M-matrix if and only if* $\rho(\mathbf{P}^{\max}) \leq \frac{1}{\lambda}$, *where* ρ *is the spectral radius.*

Proof. Let $\mathbf{M} = \mathbf{I} - \lambda\mathbf{P}$. Clearly, $\mathbf{M}$ is bounded element-wise by $\mathbf{M}^{\mathbf{max}} = \mathbf{I} - \lambda\mathbf{P}^{\mathbf{min}}$ and $\mathbf{M}^{\mathbf{min}} = \mathbf{I} - \lambda\mathbf{P}^{\mathbf{max}}$. The upper bound $\mathbf{M}^{\mathbf{max}}$ has no positive off-diagonal elements, as all of the elements of $\mathbf{P}^{\mathbf{max}}$ are between 0 and 1. So, the only thing we need to check is that $\mathbf{M}^{\mathbf{min}}$ is an M-matrix.

From the definition of an M-matrix [Berman and Plemmons, 1994], $\mathbf{M^{min}}$ is an M-matrix if and only if $1 \geq \rho(\lambda \mathbf{P^{max}}) = \lambda \rho(\mathbf{P^{max}})$. This completes the proof. $\square$

Notably, we only need check the spectral radius of the matrix formed by the upper bounds on $\mathbf{P}$. There are various conditions that can be used to obtain the necessary bound on the spectral radius. Since $\lambda < 1$, we have $1/\lambda > 1$, so if we can show that the spectral radius of the upper bound matrix is less than 1, then this automatically implies the condition in Theorem 3.4, and hence that the Gauss-Seidel method will obtain the hull, i.e., the tightest possible rectangular bounds on $\mathbf{v}$. This may occur if, for example, the row sums of the upper bounds are ≤ 1.

Example 3.5. *Let $\lambda = 0.97$, and suppose we have the following bounds on $\mathbf{P}$ and $\mathbf{r}$:*

$$\mathbf{P} : \begin{bmatrix} [0.5, 0.6] & [0.2, 0.5] \\ [0.1, 0.4] & [0.5, 0.6] \end{bmatrix}, \quad \mathbf{r} : \begin{bmatrix} [0, 100] \\ [0, 100] \end{bmatrix}$$

The bounds on $\mathbf{v}$ are:

$$\begin{array}{cc} \textit{Initial (Lemma 3.3)} & \textit{After Gauss-Seidel} \\ \begin{bmatrix} [0, 6666.67] \\ [0, 6666.67] \end{bmatrix} & \begin{bmatrix} [39.36, 6666.67] \\ [104.50, 6666.67] \end{bmatrix} \end{array}$$

Note that the spectral radius of the upper bound matrix of $\mathbf{P}$ is 1.05, which is greater than $1/\lambda = 1/0.97 \approx 1.03$. So, Gauss-Seidel will not necessarily obtain the hull, as it is not an interval M-matrix, by Theorem 3.4.

If we slightly tighten the upper bounds on $\mathbf{P}$ by changing the upper bound on $P_{2,1}$ to 0.3, then the bounds on $\mathbf{v}$ are:

$$\begin{array}{cc} \textit{Initial (Lemma 3.3)} & \textit{After Gauss-Seidel} \\ \begin{bmatrix} [0, 6666.67] \\ [0, 6666.67] \end{bmatrix} & \begin{bmatrix} [39.37, 4694.80] \\ [104.50, 3746.86] \end{bmatrix} \end{array}$$

Indeed, now the spectral radius of the upper bound matrix of $\mathbf{P}$ is 0.99, which is less than $1/\lambda = 1.03$, so by Theorem 3.4, Gauss-Seidel returns the hull. Note the tighter bounds than the prior case.

3.4.2 Solving the Final Optimization Problem

At last, we can assert the bounds found above for $\boldsymbol{\pi}, \mathbf{P}, \mathbf{r}, \mathbf{v}$ as constraints and solve the full bilinear program with any out-of-the-box solver. We summarize our procedure in Algorithm 4.

Algorithm 4 Decomposition and bound propagation scheme to optimize the total reward

```
Input: Instance of the infinite-horizon total reward optimization problem
for $i = 1$ to $k_f$, $j = 1$ to $\ell_i$ do
    Minimize and maximize $\theta_{i,j}$ over $\mathcal{X}$
end for
Propagate the bounds on $\boldsymbol{\theta}$ to $\boldsymbol{\pi}, \mathbf{P}, \mathbf{r}$ using Lemma 3.1
Obtain initial bounds on $\mathbf{v}$ using Lemma 3.3
Tighten the bounds on $\mathbf{v}$ using Gauss-Seidel
Optimize $\boldsymbol{\pi}^\top \mathbf{v}$ with an out-of-the-box solver, with obtained bounds on $\boldsymbol{\pi}, \mathbf{P}, \mathbf{r}, \mathbf{v}$ as constraints
```

Our algorithm works with any standard bilinear or MILP solver, avoiding the need for specialized branch-and-bound schemes or custom callbacks that require expert implementation. The bound propagation and Gauss-Seidel are also straightforward to implement. Crucially, our method solves the problem to global optimality if the underlying bilinear solver can, and so we do not discuss optimality guarantees further, as, e.g., Gurobi can solve bilinear problems to global optimality.

3.5 Extensions

3.5.1 Reachability, Hitting time, and Feasibility

Our results easily extend to reachability and hitting time, after handling technical details arising from the invertibility of $\mathbf{I}-\mathbf{Q}$. We provide these formulations below and then discuss how to adapt our solution algorithm to them.

Feasibility of the total infinite-horizon reward

The total infinite-horizon reward feasibility problem can be written as:

$$\begin{aligned}
\exists \mathbf{x} \quad & \text{s.t.} \\
& W_{\min} \leq \boldsymbol{\pi}^\top \mathbf{v} \leq W_{\max} \\
& \mathbf{v} = \lambda \mathbf{P}\mathbf{v} + \mathbf{r}, \\
& \boldsymbol{\theta} = \Big[\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \cdots, \boldsymbol{\theta}_{k_f}\Big]^\top, \\
& \boldsymbol{\theta}_i = f_i(\mathbf{x}) \quad \forall i \in [k_f], \\
& \boldsymbol{\pi} = \mathbf{A}_{\boldsymbol{\pi}}\boldsymbol{\theta} + \mathbf{b}_{\boldsymbol{\pi}}, \\
& \operatorname{vec}(\mathbf{P}) = \mathbf{A}_{\mathbf{P}}\boldsymbol{\theta} + \mathbf{b}_{\mathbf{P}}, \\
& \mathbf{r} = \mathbf{A}_{\mathbf{r}}\boldsymbol{\theta} + \mathbf{b}_{\mathbf{r}}, \\
& \mathbf{C}_{\boldsymbol{\pi}}\boldsymbol{\pi} \leq \mathbf{d}_{\boldsymbol{\pi}}, \\
& \mathbf{C}_{\mathbf{P}} \operatorname{vec}(\mathbf{P}) \leq \mathbf{d}_{\mathbf{P}}, \\
& \mathbf{C}_{\mathbf{r}}\mathbf{r} \leq \mathbf{d}_{\mathbf{r}}, \\
& \sum_{j=1}^{n} P_{ij} = 1 \quad \forall i \in [n], \\
& \sum_{i=1}^{n} \pi_i = 1, \\
& 0 \leq P_{ij} \leq 1 \quad \forall i,j \in [n]\times[n], \\
& 0 \leq \pi_i \leq 1 \quad \forall i \in [n], \\
& \mathbf{x} \in \mathcal{X}.
\end{aligned}$$

where $W_{\min}$ and $W_{\max}$ are bounds on the total reward, which may be $\pm\infty$.

Reachability

To reformulate the total reward formulation for reachability to a target state $S \subseteq [n]$ from a set of transient states $T \subseteq [n], T \cap S = \emptyset$, observe that we can simply make the following substitutions: $\boldsymbol{\pi}$ gets replaced with $\tilde{\boldsymbol{\pi}}$, which is the initial probability over T, $\mathbf{P}$ gets replaced with $\mathbf{Q}$, which is $\mathbf{P}$ restricted to T, and $\mathbf{r}$ gets replaced with $\mathbf{R1}$, where $\mathbf{R}$ is the transition matrix from T to S. Lastly, there is no more discount rate λ. We also require $\mathbf{Q}$ to be strictly substochastic so that it remains invertible (i.e., the row sums are strictly less than 1), and $\mathbf{R}$ and $\tilde{\boldsymbol{\pi}}$ are now substochastic (i.e., the row sums are less than or equal to 1). The optimization problem is then:

$$
\begin{aligned}
&\min/\max_{\tilde{\boldsymbol{\pi}},\mathbf{Q},\mathbf{R},\mathbf{v},\mathbf{x},\boldsymbol{\theta}} \tilde{\boldsymbol{\pi}}^\top \mathbf{v} \quad \text{s.t.} \\
&\quad \mathbf{v} = \mathbf{Q}\mathbf{v} + \mathbf{R1}, \\
&\quad \boldsymbol{\theta} = \left[\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \cdots, \boldsymbol{\theta}_{k_f}\right]^\top, \\
&\quad \boldsymbol{\theta}_i = f_i(\mathbf{x}) \quad \forall i \in [k_f], \\
&\quad \boldsymbol{\pi} = \mathbf{A}_{\boldsymbol{\pi}}\boldsymbol{\theta} + \mathbf{b}_{\boldsymbol{\pi}}, \\
&\quad \text{vec}(\mathbf{Q}) = \mathbf{A}_{\mathbf{Q}}\boldsymbol{\theta} + \mathbf{b}_{\mathbf{Q}}, \\
&\quad \text{vec}(\mathbf{R}) = \mathbf{A}_{\mathbf{R}}\boldsymbol{\theta} + \mathbf{b}_{\mathbf{R}}, \\
&\quad \mathbf{C}_{\tilde{\boldsymbol{\pi}}}\tilde{\boldsymbol{\pi}} \leq \mathbf{d}_{\tilde{\boldsymbol{\pi}}}, \\
&\quad \mathbf{C}_{\mathbf{QR}}[\text{vec}(\mathbf{Q})\,\text{vec}(\mathbf{R})] \leq \mathbf{d}_{\mathbf{QR}}, \\
&\quad \sum_{j=1}^{|T|} Q_{ij} < 1 \quad \forall i \in [|T|], \\
&\quad \sum_{j=1}^{|S|} R_{ij} \leq 1 \quad \forall i \in [|T|], \\
&\quad \sum_{i=1}^{|T|} (\tilde{\pi})_i \leq 1, \\
&\quad 0 \leq Q_{ij} < 1 \quad \forall i,j \in [|T|] \times [|T|], \\
&\quad 0 \leq R_{ij} \leq 1 \quad \forall i,j \in [|T|] \times [|S|], \\
&\quad 0 \leq (\tilde{\pi})_i \leq 1 \quad \forall i \in [|T|], \\
&\quad \mathbf{x} \in \mathcal{X}.
\end{aligned}
$$

With a slight abuse of notation, we still maintain the vector $\mathbf{v}$, but is instead now defined as $\mathbf{v} = (\mathbf{I}-\mathbf{Q})^{-1}\mathbf{R1}$. As well, note that the stochastic constraints have been replaced with substochastic constraints (strict substochastic for $\mathbf{Q}$).

The feasibility problem is:

$$
\begin{aligned}
&\exists \mathbf{x} \quad \text{s.t.}\\
&\quad W_{\min} \le \tilde{\boldsymbol{\pi}}^\top \mathbf{v} \le W_{\max}\\
&\quad \mathbf{v} = \mathbf{Q}\mathbf{v} + \mathbf{R}\mathbf{1},\\
&\quad \boldsymbol{\theta} = \left[\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \cdots, \boldsymbol{\theta}_{k_f}\right]^\top,\\
&\quad \boldsymbol{\theta}_i = f_i(\mathbf{x}) \quad \forall i \in [k_f],\\
&\quad \tilde{\boldsymbol{\pi}} = \mathbf{A}_{\tilde{\boldsymbol{\pi}}}\boldsymbol{\theta} + \mathbf{b}_{\tilde{\boldsymbol{\pi}}},\\
&\quad \operatorname{vec}(\mathbf{Q}) = \mathbf{A}_{\mathbf{Q}}\boldsymbol{\theta} + \mathbf{b}_{\mathbf{Q}},\\
&\quad \operatorname{vec}(\mathbf{R}) = \mathbf{A}_{\mathbf{R}}\boldsymbol{\theta} + \mathbf{b}_{\mathbf{R}},\\
&\quad \mathbf{C}_{\tilde{\boldsymbol{\pi}}}\tilde{\boldsymbol{\pi}} \le \mathbf{d}_{\tilde{\boldsymbol{\pi}}},\\
&\quad \mathbf{C}_{\mathbf{QR}}[\operatorname{vec}(\mathbf{Q})\,\operatorname{vec}(\mathbf{R})] \le \mathbf{d}_{\mathbf{QR}},\\
&\quad \sum_{j=1}^{|T|} Q_{ij} < 1 \quad \forall i \in [|T|],\\
&\quad \sum_{j=1}^{|S|} R_{ij} \le 1 \quad \forall i \in [|T|],\\
&\quad \sum_{i=1}^{|T|} \tilde{\pi}_i \le 1,\\
&\quad 0 \le Q_{ij} < 1 \quad \forall i,j \in [|T|] \times [|T|],\\
&\quad 0 \le R_{ij} \le 1 \quad \forall i,j \in [|T|] \times [|S|],\\
&\quad 0 \le \tilde{\pi}_i \le 1 \quad \forall i \in [|T|],\\
&\quad \mathbf{x} \in \mathcal{X}.
\end{aligned}
$$

Hitting time

The hitting time to a target set $S \subseteq [n]$ from a set of transient states $T \subseteq [n], T \cap S = \emptyset$ formulation is extremely similar to the reachability formulation, except that we replace $\mathbf{R1}$ with $\mathbf{1}$. Hence, $\mathbf{R}$ is no longer a decision variable. Also, now, $\mathbf{v}$ is defined as $\mathbf{v} = (\mathbf{I} - \mathbf{Q})^{-1}\mathbf{1}$. Note again the substochastic constraint on $\tilde{\boldsymbol{\pi}}$ and the strict substochastic constraint on $\mathbf{Q}$. The optimization problem is:

$$
\begin{aligned}
&\min/\max_{\tilde{\boldsymbol{\pi}},\mathbf{Q},\mathbf{v},\mathbf{x},\boldsymbol{\theta}} \tilde{\boldsymbol{\pi}}^\top \mathbf{v} \quad \text{s.t.}\\
&\quad \mathbf{v} = \mathbf{Q}\mathbf{v} + \mathbf{1},\\
&\quad \boldsymbol{\theta} = \left[\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \cdots, \boldsymbol{\theta}_{k_f}\right]^\top,\\
&\quad \boldsymbol{\theta}_i = f_i(\mathbf{x}) \quad \forall i \in [k_f],\\
&\quad \tilde{\boldsymbol{\pi}} = \mathbf{A}_{\tilde{\boldsymbol{\pi}}}\boldsymbol{\theta} + \mathbf{b}_{\tilde{\boldsymbol{\pi}}},\\
&\quad \operatorname{vec}(\mathbf{Q}) = \mathbf{A}_{\mathbf{Q}}\boldsymbol{\theta} + \mathbf{b}_{\mathbf{Q}},\\
&\quad \mathbf{C}_{\tilde{\boldsymbol{\pi}}}\tilde{\boldsymbol{\pi}} \leq \mathbf{d}_{\tilde{\boldsymbol{\pi}}},\\
&\quad \mathbf{C}_{\mathbf{Q}} \operatorname{vec}(\mathbf{Q}) \leq \mathbf{d}_{\mathbf{Q}},\\
&\quad \sum_{j=1}^{|T|} Q_{ij} < 1 \quad \forall i \in [|T|],\\
&\quad \sum_{i=1}^{|T|} \tilde{\pi}_i \leq 1,\\
&\quad 0 \leq Q_{ij} < 1 \quad \forall i,j \in [|T|] \times [|T|],\\
&\quad 0 \leq \tilde{\pi}_i \leq 1 \quad \forall i \in [|T|],\\
&\quad \mathbf{x} \in \mathcal{X}.
\end{aligned}
$$

The feasibility problem is:

$$
\begin{aligned}
&\exists \mathbf{x} \quad \text{s.t.} \\
&\quad W_{\min} \leq \tilde{\boldsymbol{\pi}}^\top \mathbf{v} \leq W_{\max} \\
&\quad \mathbf{v} = \mathbf{Q}\mathbf{v} + \mathbf{1}, \\
&\quad \boldsymbol{\theta} = \left[\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \cdots, \boldsymbol{\theta}_{k_f}\right]^\top, \\
&\quad \boldsymbol{\theta}_i = f_i(\mathbf{x}) \quad \forall i \in [k_f], \\
&\quad \tilde{\boldsymbol{\pi}} = \mathbf{A}_{\tilde{\boldsymbol{\pi}}}\boldsymbol{\theta} + \mathbf{b}_{\tilde{\boldsymbol{\pi}}}, \\
&\quad \operatorname{vec}(\mathbf{Q}) = \mathbf{A}_{\mathbf{Q}}\boldsymbol{\theta} + \mathbf{b}_{\mathbf{Q}}, \\
&\quad \mathbf{C}_{\tilde{\boldsymbol{\pi}}}\tilde{\boldsymbol{\pi}} \leq \mathbf{d}_{\tilde{\boldsymbol{\pi}}}, \\
&\quad \mathbf{C}_{\mathbf{Q}} \operatorname{vec}(\mathbf{Q}) \leq \mathbf{d}_{\mathbf{Q}}, \\
&\quad \sum_{j=1}^{|T|} Q_{ij} < 1 \quad \forall i \in [|T|], \\
&\quad \sum_{i=1}^{|T|} (\pi_T)_i \leq 1, \\
&\quad 0 \leq Q_{ij} < 1 \quad \forall i,j \in [|T|] \times [|T|], \\
&\quad 0 \leq \tilde{\pi}_i \leq 1 \quad \forall i \in [|T|], \\
&\quad \mathbf{x} \in \mathcal{X}.
\end{aligned}
$$

Solving

For reachability and hitting time, the key difference that affects our theoretical results and algorithms is that the matrix $\mathbf{Q}$, which is a submatrix of $\mathbf{P}$ restricted to the transient states, is strictly substochastic, and we don't have a discount factor λ anymore. This introduces some technical difficulties in guaranteeing invertibility of $\mathbf{I} - \mathbf{Q}$. However, all our theoretical results all follow through with minor modifications.

Lemma 3.6. *For a strictly row-substochastic matrix $\mathbf{Q}$ with all elements in $[0, 1]$:*

1. *The spectral radius of $\mathbf{Q}$ is strictly less than 1.*
2. *The matrix $(\mathbf{I} - \mathbf{Q})$ is a (non-singular) M-matrix.*
3. *The matrix $(\mathbf{I} - \mathbf{Q})^{-1}$ has all non-negative elements.*
4. *The row sums of $(\mathbf{I} - \mathbf{Q})^{-1}$ are at most $1/(1-\alpha)$, where $\alpha = \max_i \sum_j Q_{ij}$ is the maximum row sum of $\mathbf{Q}$.*
5. *Each element of $(\mathbf{I} - \mathbf{Q})^{-1}$ is in the interval $[0, 1/(1-\alpha)]$.*

Proof. The row sum of $\mathbf{Q}$ is strictly less than 1, so, by a well-known result in linear algebra, the spectral radius of $\mathbf{Q}$ is strictly less than 1. It follows that $\mathbf{I} - \mathbf{Q}$ is a (non-singular) M-matrix by the definition of an M-matrix [Berman and Plemmons, 1994], and hence that the inverse has all non-negative elements by Theorem 2.3 in Chapter 6 of Berman and Plemmons [1994]. □

Note that the maximum row sum α is strictly less than 1, so the formula is well-defined. We use this result to obtain the appropriate initial bounds on $\mathbf{v}$ for the reachability and hitting time formulations.

Corollary 3.7. *Given a Markov reachability problem with element-wise bounds on* $\mathbf{Q}$ *and* $\mathbf{R}$*, as defined in Section 3.5, the initial bounds on* $\mathbf{v} = (\mathbf{I} - \mathbf{Q})^{-1}\mathbf{R}\mathbf{1}$ *are:*

$$0 \leq v_i \leq \frac{1}{1-\alpha} \max_j \sum_k R_{jk}^{\max}$$

where $\alpha = \max_i \sum_j Q_{ij}^{\max}$ *is the maximum row sum of* $\mathbf{Q}$*.*

Proof. First, note that the vector $\mathbf{R1}$ has elements which are the row sums of $\mathbf{R}$. So, for each i, $\sum_j R_{ij}^{\min} \leq (\mathbf{R1})_i \leq \sum_j R_{ij}^{\max}$. Next, noting that the row sums of $(\mathbf{I} - \mathbf{Q})^{-1}$ are at most $1/(1-\alpha)$, we have, by a similar argument as in the proof of Lemma 3.3, that the upper bound on v_i is:

$$v_i \leq \frac{1}{1-\alpha} \max_j \sum_k R_{jk}^{\max}$$

For the lower bound, unlike the argument in Lemma 3.3, we don't know what the row sums of $(\mathbf{I} - \mathbf{Q})^{-1}$ actually are, so we use the smallest possible row sum, which is 0. So, we have:

$$v_i \geq 0$$

□

Corollary 3.8. *Given a Markov hitting time problem with element-wise bounds on* $\mathbf{Q}$*, as defined in Section 3.5, the initial bounds on* $\mathbf{v} = (\mathbf{I} - \mathbf{Q})^{-1}\mathbf{1}$ *are:*

$$0 \leq v_i \leq \frac{1}{1-\alpha}$$

where $\alpha = \max_i \sum_j Q_{ij}^{\max}$ *is the maximum row sum of* $\mathbf{Q}$*.*

Proof. The bound simply follows from the fact that $\mathbf{v}$ is the row sums of $(\mathbf{I} - \mathbf{Q})^{-1}$. □

Note, however, that we *do not* know what the row sums of $\mathbf{Q}$ are, because it is not fixed. However, in practice, in order to assert the strict row-substochastic property, we use a small numerical offset, which is what we use in our implementation. Specifically, we set the row sums of $\mathbf{Q}$ to be at most $1 - \epsilon$, where ϵ is by default set to 10^{-6} in our software implementation. Based on this, we can set α to be $1 - \epsilon$.

Next, we can tighten the bounds on $\mathbf{v}$ by forming the appropriate linear systems: $(\mathbf{I} - \mathbf{Q})\mathbf{v} = \mathbf{R1}$ for reachability and $(\mathbf{I} - \mathbf{Q})\mathbf{v} = \mathbf{1}$ for hitting time. In both cases, though $\mathbf{Q}$ is a submatrix of $\mathbf{P}$, it is square, so we can analyze the similar interval M-matrix condition for the Gauss-Seidel method to achieve the hull of the solution set.

Corollary 3.9. *Let* $\mathbf{Q}$ *be a row-substochastic matrix bounded element-wise by* $\mathbf{Q}^{\mathbf{min}}$ *and* $\mathbf{Q}^{\mathbf{max}}$*. Then, it is an interval M-matrix if and only if* $\rho(\mathbf{Q}^{\mathbf{max}}) \leq 1$*.*

Proof. The proof is identical to that of Theorem 3.4, but without the discount factor. □

Based on the above extensions of our theoretical results, we now have similar algorithms for reachability and hitting time. As usual, we first optimize for bounds on $\boldsymbol{\theta}$, then we propagate to $\mathbf{Q}$ and $\mathbf{R}$ (if applicable), obtain initial bounds on $\mathbf{v}$ as above, and then apply the Gauss-Seidel method to tighten the bounds on $\mathbf{v}$.

Our method easily extends to the feasibility version of all three problems. We still obtain all the necessary bounds as in the optimization version. At the end, when we would optimize for the quantity of interest, we simply check if the overall problem is feasible, given the bounds on the quantity of interest.

3.5.2 Special Cases

If one or more of $\boldsymbol{\pi}, \mathbf{P}, \mathbf{r}$ are fixed, then the problem may become much easier to solve. For example, if $\mathbf{P}$ and $\mathbf{r}$ are fixed, then $\mathbf{v}$ is completely determined as the solution to a linear system, so we can eliminate the Bellman constraint, and the objective becomes simply linear in π. We enumerate all of these possibilities below.

Table 3.1: Special Cases for Optimization Problem. Objective refers to the objective $\boldsymbol{\pi}^\top \mathbf{v}$ and constraint refers to the Bellman constraint $\mathbf{v} = \lambda \mathbf{P}\mathbf{v} + \mathbf{r}$.

Fixed Variable(s)	Objective	Constraint	Explanation
$\boldsymbol{\pi}$	Linear	Bilinear	Only objective affected
$\mathbf{P}$	Bilinear	Linear	Only constraint affected
$\mathbf{r}$	Bilinear	Bilinear	Constraint affected but bilinear term remains
$\boldsymbol{\pi}, \mathbf{P}$	Linear	Linear	Objective and constraint affected
$\boldsymbol{\pi}, \mathbf{r}$	Linear	Bilinear	Objective and constraint affected, but bilinear term remains
$\mathbf{P}, \mathbf{r}$	Linear	Eliminated	$\mathbf{v}$ fully determined, leaving objective linear in $\boldsymbol{\pi}$

3.6 Software: `markovml`

We developed the Python package `markovml` to specify Markov chains or reward processes with embedded pretrained machine learning models. It is available at `https://github.com/mmaaz-git/markovml`. In this section, we describe aspects of its implementation and provide a walk-through of how it can be used.

Most notably, it provides a domain-specific language lets users: (1) instantiate a Markov process, (2) add pretrained ML models from `sklearn` [Pedregosa et al., 2011] and PyTorch [Paszke et al., 2019], (3) link model outputs to Markov parameters with affine equalities, (4) include extra linear inequalities, (5) specify the feature set with MILP constraints, and (6) optimize reachability, hitting time, or total reward.

Our package is built on the optimization solver Gurobi, specifically its Python interface, `gurobipy` [Gurobi Optimization, LLC, 2024]. As the user sets up their model, our software constructs, in the background, the equivalent formulation in Gurobi. We also then directly leverage Gurobi's ability to solve bilinear programs to global optimality (our decomposition scheme also calls out to Gurobi's solver).

3.6.1 Supported Models

Our package supports a variety of regression and classification models, including linear, tree-based, and neural networks. For some models, the MILP formulation is provided by `gurobi-machinelearning` [Gurobi Optimization, 2024], a Python package built on top of `gurobipy` which embeds pretrained ML models into a Gurobi model, while for others, we implemented the MILP formulation ourselves.

We support the following models from `sklearn`:

- `LinearRegression`
- `Ridge`
- `Lasso`
- `LogisticRegression`
- `DecisionTreeRegressor`
- `DecisionTreeClassifier`
- `RandomForestRegressor`
- `RandomForestClassifier`
- `MLPRegressor`
- `MLPClassifier`

From `pytorch`, we support neural networks built as `nn.Sequential` models with ReLU or linear layers, with possibly a softmax layer at the end for classifiers.

Lastly, we implemented a model called `DecisionRules`, which allows the user to specify a series of "if-then" rules specified in natural language, e.g., "if age $\geq$ 65 then 0.8". This enables encoding tables from the literature, like age-stratified mortalities, common in healthcare.

We make use of the MILP formulations from `gurobi-machinelearning`. At time of writing, `gurobi-machinelearning` did not support the softmax function, although it did support the logistic function. Hence, for multilayer perceptron classifiers from `sklearn` and for neural network classifiers from `torch`, we implemented the MILP formulation ourselves. We also implemented the MILP formulation of our `DecisionRules` model through a series of logical implications.

3.6.2 Using `markovml`

We walk through how to use `markovml`. Our software comes bundled with extensive documentation and tutorials.

Instantiating a Markov process

There are three objects in the `markovml` package: `MarkovReward`, `MarkovReach`, and `MarkovHitting`. They correspond to the problem you are trying to solve: verifying the total reward, reachability, and hitting time, respectively. They have slightly different interfaces and internal operations, but there is lots of common functionality, so it is easy to switch between them (in fact, they all inherit from the same base class, `AbstractMarkov`, which provides almost all of the functionality). While they have common setups, there are specific elements to each of them; for example, `MarkovReward` requires the setting of a reward vector **r** while the other do not. Below are some examples of instantiating each of these objects.

```
mrp = MarkovReward(n_states=2, n_features=2)
m_reach = MarkovReach(n_states=2, n_features=2, n_transient=1, n_targets=1)
m_hitting = MarkovHitting(n_states=2, n_features=2, n_transient=1)
```

Adding ML models

The next step is to add a machine learning model to the problem. This is done with the `add_ml_model` function. This function takes a pretrained model as input and embeds its MILP formulation in the underlying optimization problem. Suppose we have pretrained models `reward_model`, `trans_model`, `reward_model2`, `trans_model2`, where the reward models are regression models and the transition models are classification models. We can add them to the underlying Markov process as:

```
mrp.add_ml_model(reward_model)
mrp.add_ml_model(trans_model)
mrp.add_ml_model(reward_model2)
mrp.add_ml_model(trans_model2)
```

We can then access the variables corresponding to the *outputs* of each of these models using the `ml_outputs` attribute of the Markov object; e.g., `mrp.ml_outputs[0][0]` is the first output of the first added model (`reward_model`).

Setting parameters

There are three ways to set the parameters of the Markov process: (1) pass constants at initialization, (2) use `set_to_const` or `set_to_ml_output`, or (3) use the setting helper functions.

If it is known that an entire vector or matrix is a constant, then this can be passed at initialization, as below.

```
MarkovReward(n_states=2, n_features=2, r=[1, 0])
```

The `set_to_const` or `set_to_ml_output` functions can be used to set elements one at a time either to a constant or to an affine function of one of the ML models' outputs, as below.

```
mrp.set_to_const(mrp.r[0], 1)
mrp.set_to_const(mrp.r[1], 0)

mrp.set_to_ml_output(mrp.r[2], mrp.ml_outputs[0][0])
mrp.set_to_ml_output(mrp.r[3], 2*mrp.ml_outputs[0][0]-1)
```

Lastly, each of the classes have setting helper functions which can be used to set the parameters altogether, as it may be cumbersome to set one at a time. Depending on the class, these are: `set_pi`, `set_P`, `set_Q`, `set_r`, `set_R`.

```
mrp.set_pi([mrp.ml_outputs[1][0],
1 - mrp.ml_outputs[1][0],
0, 0, 0])
```

Adding linear constraints on parameters

We can add linear constraints on the parameters quite simply:

```
mrp.add_constraint(mrp.r[0] >= 1)
```

Defining the feature space

To define $\mathcal{X}$, we use `add_feature_constraint` and `add_feature_aux_variable`. The first function allows adding linear inequalities, and the second allows adding auxiliary continuous or discrete variables. Together, these can encode any MILP-representable set.

```
mrp.add_feature_constraint(mrp.features[0] + mrp.features[1] <= 1.5)
mrp.add_feature_constraint(mrp.features[2] >= mrp.features[3])
```

Optimizing

Once all parameters have been set and the feature space has been fixed, we can optimize our metric of interest quite simply.

```
mrp.optimize(sense="max")
```

It is possible to pass various solving options here as well as find a feasible solution instead of optimizing.

Complete example

In only a few lines, a user can build a Markov process, integrate an ML model, and optimize the reward.

```
from markovml.markovml import MarkovReward
from sklearn.linear_model import LogisticRegression
import numpy as np

# Create a Markov reward process
mrp = MarkovReward(n_states=2, n_features=2)

# fix some parameters
mrp.set_r([1, 0])
mrp.set_pi([1, 0])

# train a classifier
X = np.random.rand(100, 2)
y = np.random.randint(0, 2, 100)
clf = LogisticRegression().fit(X, y)

# add classifier
mrp.add_ml_model(clf)

# link to transitions
mrp.set_P([[1 - mrp.ml_outputs[0][0], mrp.ml_outputs[0][0]], [0, 1]])

# define feature space
```

```
    mrp.add_feature_constraint(mrp.features[0] >= 65)
    mrp.add_feature_constraint(mrp.features[1] >= 100)

    # optimize!
    mrp.optimize(sense="max")
```

Decision rules

The following code demonstrates building a `DecisionRules` model, which allows building "if-then" rules in a natural language syntax.

```
    dr = DecisionRules(features = ['age', 'income'])
    dr.fit(rules = [
    "if age > 20 then 2.5",
    "if income >= 50000 and age < 30 then -1.0",
    "else 0.0"
    ])
```

It is then possible to "predict" using such a model on new data. This can also be added into a Markov object, just as any other learned model, and we implemented the MILP formulation through a series of logical constraints.

3.7 Numerical Experiments

We compare our method to solving the optimization problem directly with an out-of-the-box solver. Our experiments will analyze the following drivers of complexity of the bilinear program: the number of states, the number of ML models, and the model complexity, notably for trees and neural networks.

3.7.1 Setup

For our experiments, we generate Markov chains and train ML models on randomly generated data, and compare the solution time of our method versus directly with Gurobi.

For all experiments, we fix the number of features $m = 5$, the feature set $\mathcal{X} = [-1,1]^5$, and the discount factor $\lambda = 0.97$. We train ML models on randomly generated data (10,000 random points). For regression models, we draw $\mathbf{X} \sim \mathcal{N}(0,1)^{10000\times 5}$, $\boldsymbol{\beta} \sim \mathcal{N}(0,1)^5$, and then generate data points as $\mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(0,1)^{10000}$. For classification models, we use the same linear form but draw data points from a Bernoulli distribution with probabilities $\sigma(\mathbf{X}\boldsymbol{\beta})$, where $\sigma(\cdot)$ is the logistic function.

For rewards, we train a regression model whose output is set to r_1. For $i = 2, \dots, n$, we set r_i to be output divided by i. For probabilities, we train binary classifiers for $\boldsymbol{\pi}$ and rows of $\mathbf{P}$. For $\boldsymbol{\pi}$, we set π_1 equal to the classifier output, the remaining probability is assigned to π_2, and the remaining are set to zero. For rows of $\mathbf{P}$, the classifier output is the probability of remaining in the same state, with the remaining probability assigned to transitioning to the next state, and the rest of the row is 0. We make the last state absorbing, i.e., transitions to itself with probability 1. The Markov process is essentially a discrete-time birth process, as one can only transition to the next state, or stay in the same state, with the final state being absorbing.

For each configuration, we run 10 instances, each with random data, train the models on the random data, and solve the optimization problem directly with an out-of-the-box solver and with our method.

We perform the following experiments:

1. *State Space:* Vary $n \in \{5, 10, 20, 50, 100, 200\}$ using three models: a linear regression for $\mathbf{r}$ and two logistic regressions for $\boldsymbol{\pi}$ and the first row of $\mathbf{P}$ (the remaining rows are uniform).

2. *Model Count:* Fix $n = 20$ with linear regression for $\mathbf{r}$ and logistic regression for all probability models. Models for $\mathbf{r}$ and $\boldsymbol{\pi}$ are always used, while the number of modeled rows in $\mathbf{P}$ varies from 1 to 19, so that the total number of models runs from 3 to 21.

3. *Decision Tree Depth:* Fix $n = 20$ using three decision tree models (for $\mathbf{r}$, $\boldsymbol{\pi}$, and the first row of $\mathbf{P}$) with depths in $\{2, 3, 4, 6, 8\}$. Remaining rows of $\mathbf{P}$ are uniform.

4. *Neural Network Architecture:* Fix $n = 20$ using three ReLU multilayer perceptrons, for $\mathbf{r}$, $\boldsymbol{\pi}$, and the first row of $\mathbf{P}$. We vary hidden layers in $\{1, 2\}$ and neurons per layer in $\{5, 10, 15, 20\}$, with remaining rows of $\mathbf{P}$ set uniformly.

We record runtime, objective value, and optimizer status. For our method, the runtime includes solving the smaller MILPs, bound computations, the interval Gauss-Seidel phase, and the final optimization. All experiments are run on an Intel Core i7 with 6 cores using Gurobi Optimizer 12.0.0 [Gurobi Optimization, LLC, 2024] (1200 second limit per problem, and presolve disabled throughout). All code and instructions on how to reproduce our experiments is included with our code.

We computed the geometric mean and geometric standard deviations of runtimes, and proportions of optimizer statuses. For the runtime analysis, we only kept instances that were solved to optimality. Sometimes, the optimizer would return a suboptimal status, meaning it cannot prove optimality. If for an instance, our method did converge to optimality, and the direct method returned a suboptimal status, and the objective values were the same up to a tolerance of 10^{-12}, we consider the suboptimal status to be optimal for the runtime statistical analysis. We performed paired t-tests on the log of runtimes (for instances where both methods returned optimal) and χ^2 tests for statuses.

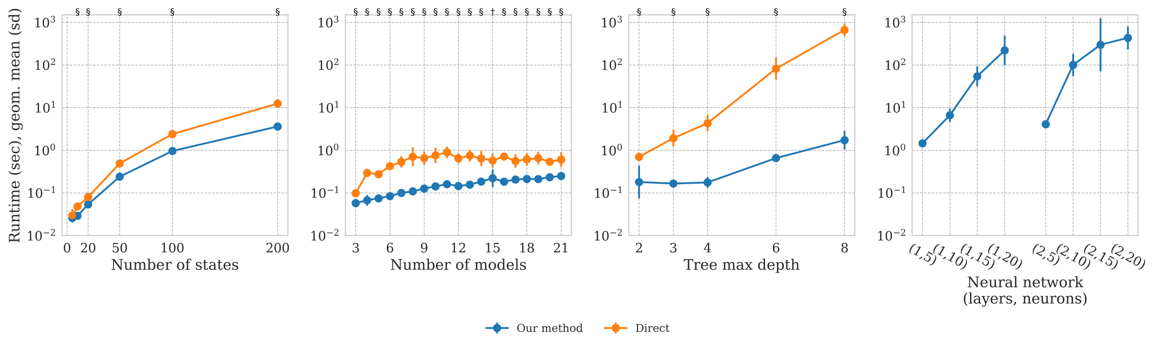


Figure 3.2: Runtimes of our method versus direct solving. Each panel shows results from experiments 1-4 (left to right). Points represent geometric means with error bars indicating standard deviation. Statistical significance from paired t-tests shown above: * ($p < 0.05$), † ($p < 0.01$), and § ($p < 0.001$).

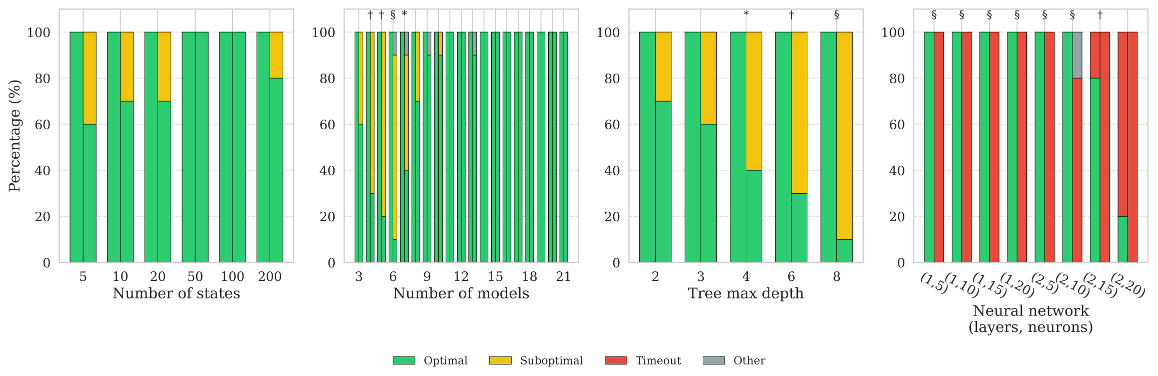


Figure 3.3: Proportion (%) of instances categorized as optimal, suboptimal, timed out, or other. Each panel shows experiments 1-4 (left to right). Our method is the left bar and direct method is the right bar. Statistical significance from χ^2 shown above: * ($p < 0.05$), † ($p < 0.01$), and § ($p < 0.001$).

Table 3.2: Geometric means of runtimes and number of instances solved for ablations of our method. Ablation numbers refer to ablating: (1) finding bounds on $\boldsymbol{\theta}$, (2) propagating bounds to $\boldsymbol{\pi}, \mathbf{P}, \mathbf{r}$, (3) initial bounds on $\mathbf{v}$, and (4) tightening bounds on $\mathbf{v}$. Lastly, None refers to no ablations. Note that ablating (1) is equivalent to directly solving, and None is equivalent to our full method.

	(1)	(2)	(3)	(4)	None
Runtime (sec), geom. mean (sd)	710.79 (1.22)	292.36 (1.64)	282.42 (1.59)	1.85 (1.55)	1.89 (1.56)
Instances solved	6/10	8/10	7/10	10/10	10/10

3.7.2 Results

Figure 3.2 shows that our method consistently outperforms direct solving and scales much better with all measures of complexity, and the differences are statistically significant, often at 1% and 0.1% significance. This is despite our method involving multiple optimization problems.

The difference is particularly striking as models get more complex: in the tree depth experiments, our method is on average 117x faster for trees of depth 6 (direct mean 82.1 sec, our method mean 0.7 sec), and 391x faster for depth 8 (direct mean 665.3 sec, our method mean 1.7 sec) – reaching a 1000x speedup on some instances. For neural networks, solving the problem directly times out in every instance, as seen in Figure 3.3, even for the smallest architecture with 1 hidden layer of 5 neurons. However, our method solves all instances with 1 hidden layer and most instances with 2 hidden layers, although it often times out with 2 hidden layers of 20 neurons each. Figure 3.3 also shows that the direct method often struggles with proving optimality.

In order to isolate the effects of different parts of our method, we performed an ablation study. We redid the decision trees experiment fixing a tree depth of 8. We ablated, in turn: (1) finding bounds on $\boldsymbol{\theta}$, (2) propagating bounds to $\boldsymbol{\pi}, \mathbf{P}, \mathbf{r}$, (3) initial bounds on $\mathbf{v}$, and (4) tightening bounds on $\mathbf{v}$. Ablating a step means ablating it and all subsequent steps. As shown in Table 3.2, the biggest contributor to our method's speedup is obtaining the initial bounds on $\mathbf{v}$, which causes a 153x speedup on average compared to ablating it.

3.8 Discussion and Conclusion

We have introduced a novel framework for the formal verification of Markov processes with learned parameters, enabling exact reasoning about systems where transition probabilities and rewards are specified by machine learning models. Our approach formulates the verification task as a bilinear program and introduces a decomposition and bound propagation scheme that significantly accelerates computation—often by orders of magnitude, while maintaining global optimality guarantees. This work contributes to the broader goal of safe and transparent deployment of ML in high-stakes settings. Our healthcare case study demonstrates how the method can support robust decision making in critical domains. We have implemented this in the software package `markovml`, which provides a flexible language for defining Markov processes, embedding a range of ML models, and optimizing key metrics.

Limitations Our experiments show that the method scales well across problem dimensions including the number of states, models, and model complexity. However, a key limitation is scalability for neural networks: our method worked well on small networks but degrades with increasing depth or width. This is consistent with known computational hardness results for neural network verification. Future work could improve performance by integrating state-of-the-art techniques from the neural network verification literature, such as α, β-CROWN. One of the key limitations in the real-world use of this method is that it assumes that the underlying ML model is accurate and well-fit. In general, formal verification takes the ML model as is, and does not address issues related to, e.g., how errors in the ML model would propagate to uncertainty in the verification results. This issue is perhaps outside of the scope of this paper but is an important caveat of which users should be aware.

Future directions Our software package, `markovml`, is open-source and designed for extensibility. We envision future extensions that support richer ML models and eventually integrates with other neural network verifiers. Our current set-up assumes that the feature vector $\mathbf{x}$ lies in a MILP-representable set $\mathcal{X}$. However, this may lead to optimizing over a set that contains an unrealistic combinations of features. A rich extension would be to assign a probability distribution to $\mathcal{X}$. By analogy, our present work is similar to a robust optimization framework while assigning a probability distribution would be similar to a distributionally robust optimization framework. We hope this work encourages further research at the intersection of probabilistic model checking and formal verification of ML.

Chapter 4

Cost-Effectiveness of Drone-Delivered Automated External Defibrillators for Cardiac Arrest

4.1 Introduction

Out-of-hospital cardiac arrest (OHCA) affects approximately 400,000 adults in the US and Canada each year, with survival rates generally below 10% [McCormick, 1976, Heart and Stroke Foundation of Canada, 2023]. Rapid treatment with cardiopulmonary resuscitation (CPR) and defibrillation with an automated external defibrillator (AED) is crucial for survival [Berger, 2020].

Recently, there has been rapidly growing interest in using unmanned aerial vehicles, or drones, to deliver AEDs to cardiac arrest victims. Several computational studies have optimized the design of drone networks and found that drone-delivered AEDs can theoretically arrive much faster than typical ambulance response times [Boutilier et al., 2017, Zègre-Hemsey et al., 2018, Pulver et al., 2016, Pulver and Wei, 2018, Boutilier and Chan, 2022]. Others have studied the feasibility of drone delivery by performing test flights to simulated cardiac arrests, validating the potential of rapid AED delivery [Cheskes et al., 2020, Zègre-Hemsey et al., 2020, Claesson et al., 2016]. Finally, an ongoing study in Sweden reported over 90% success in AED delivery to real cardiac arrests [Schierbeck et al., 2022a] which led to the first recorded save of a sudden cardiac arrest by drone [Schierbeck et al., 2022b, 2023].

Despite these promising results, the cost-effectiveness of AED drone delivery has not been rigorously established. Existing studies were restricted to population-based estimates of cardiac arrest incidence and, most importantly, excluded healthcare costs in their calculations, which are much greater than drone technology costs [Bogle et al., 2019, Bauer et al., 2021, Röper et al., 2023]. Since many jurisdictions are developing and testing drone networks for AED delivery, a comprehensive analysis is needed. In this paper, we conduct a rigorous cost-effectiveness analysis using a 10-year cohort of OHCA data, along with in-hospital and post-discharge healthcare costs, from Ontario, Canada. We evaluate close to one thousand different drone networks, develop statistical models to predict patient outcomes under drone response times, and simulate patient trajectories post-arrest using a Markov model.

4.2 Summary of Methods

This study was approved by the Research Ethics Board of Unity Health Toronto (protocol #: 18-091) and was conducted in accordance with the CHEERS reporting guidelines.

4.2.1 Data Source and Study Setting

We obtained OHCA data from the Toronto, Canada site of the Resuscitation Outcomes Consortium (ROC) cardiac arrest registry. From 2006-2015, ROC collected OHCA patient, event, response, and outcome characteristics for 10 sites across the United States and Canada [Lin et al., 2011]. The Toronto site included participation from eight EMS agencies19, collectively covering over 7 million people over 26,000 square km.

4.2.2 Study Population

We included all EMS-treated OHCAs recorded by the Toronto ROC site between Jan. 1, 2006 and Dec. 31, 2015. Patients under age 18, as well as those missing incident location, EMS response timestamps, or OHCA outcome were excluded (additional details in Section 4.3.1).

4.2.3 Drone Network Optimization

The 538 EMS, fire, and police stations in the study region were considered as potential drone bases. Based on current technology and implementation considerations, at most one drone is placed at each base. The locations of drones were determined using mathematical optimization (additional details in Section 4.3.2). To generate a large and diverse set of possible drone networks, we used three different types of objectives:

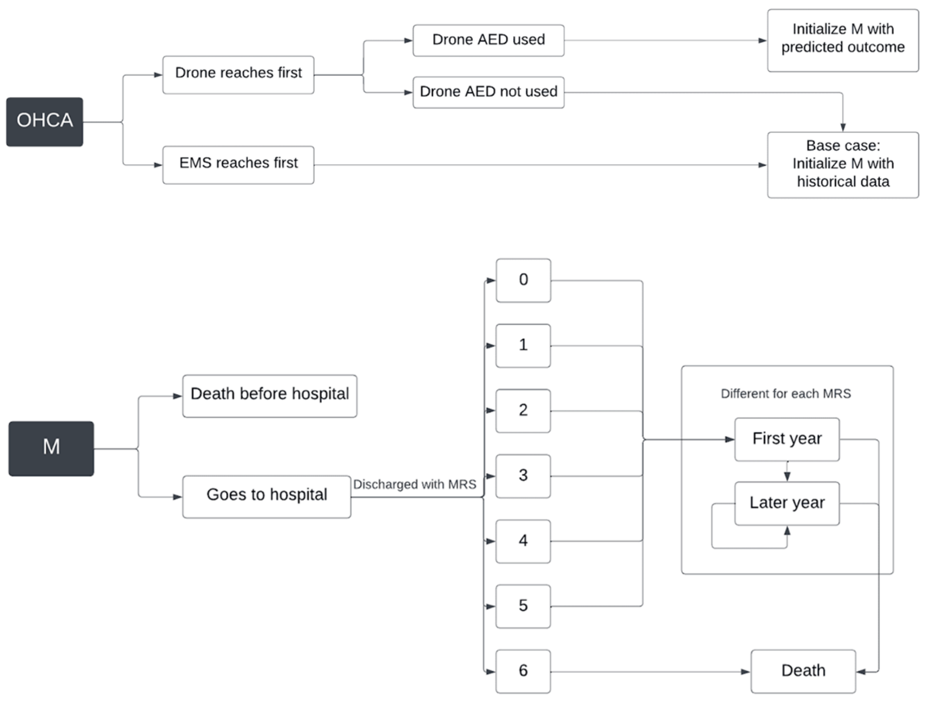
1. Coverage: Reach as many people as possible within 1, 2, 3, 4, 5, 6, or 7 minutes
2. Mean: Minimize the mean response time
3. 90th percentile: Minimize the 90th percentile, by minimizing the mean of the upper 10% of the response time distribution

For each objective, we create network sizes (i.e., number of drones) ranging from 5 to 535 drones, in increments of 5, plus a network with 538 drones (i.e., all candidate bases). In total, we generated 964 different drone networks (107 network sizes for each of the nine objectives, plus the network of 538).

4.2.4 Drone Specifications

We utilized drone specifications obtained through discussions with InDro Robotics (Victoria, British Columbia, Canada) for drones being considered in Ontario, and Everdrone (Sweden) who supply

Figure 4.1: Model used to chart trajectory of patient after an OHCA with a drone intervention. The M indicates the Markov microsimulation model, with a cycle length of one year



the drones being used in Sweden [Schierbeck et al., 2023] (see Table 4.1). The cruising altitude of the drone was 30 m. Its cruising speed was 56 km/h. Upon reaching its destination, the AED is lowered by a tether. The combined time for takeoff and lowering was 30 s.

4.2.5 Decision Model

Our decision model consisted of three components: a decision tree, several prediction models, and a Markov microsimulation model. The decision tree considered whether a drone-delivered AED would be applied to an OHCA (Figure 4.1). The base case is the status quo of no drones, for which we use the historical ambulance response time (i.e., from dispatch to arrival). For each drone network, the drone response time (i.e., dispatch to AED on ground) was calculated based on the closest drone to the OHCA. If the drone arrived first, a bystander would apply the AED with a certain probability (Table 4.1). If the drone AED was not used, either because the drone arrived after the ambulance or the bystander did not apply the AED, then we reverted to the base case for that OHCA.

The primary clinical outcome was the patient's modified Rankin Scale (mRS) value, a scale of neurological disability from 0 (no symptoms) to 6 (death) [Wilson et al., 2002]. In the base case, the historical mRS was used. In the drone case, we used three prediction models to estimate mRS: (1) probability of a shockable rhythm (binary logistic regression), (2) probability of survival to hospital admission (binary logistic regression), and (3) probability of each mRS value at hospital discharge, conditional on hospital admission (gradient boosted tree model). All models adjusted for age, sex, witnessed status, and the response time. The latter two models also adjusted for presence of a shockable heart rhythm (i.e., ventricular fibrillation or pulseless ventricular tachycardia [Sasson et al., 2010]). Since probability of a shockable rhythm degrades over time, the second and third models used the predicted shockability from the first model. More details about model training and hyperparameters are in Section 4.3.3.

Finally, the Markov microsimulation model (Figure 4.1) tracked outcomes of each patient following hospital discharge based on their actual mRS (in the base case) or estimated mRS (in the drone case), with a cycle length of one year. The model's parameters were mRS-specific (Table 4.1; Section 4.3.4).

4.2.6 Outcomes

We measured operational and healthcare costs. Operational costs comprised the cost to implement and maintain the drone network for 10 years (the length of the study's inclusion period), which were obtained through discussions with InDro Robotics and Everdrone (Table 4.1; Section 4.3.4). Healthcare costs included both in-hospital and post-discharge costs. A discount rate of 3% per year was used. Costs were expressed in 2023 Canadian dollars.

For each drone network and for the base case, we calculated the number of survivors to hospital admission, number of survivors to hospital discharge, and number of survivors with a neurologically favorable outcome (i.e., mRS 0-2). We also calculated total costs and total quality-adjusted life-years (QALYs). We identified non-dominated drone networks (based on strict and extended dominance [Gold, 1996]) and calculated incremental cost-effectiveness ratios (ICER) and net monetary benefits (NMB), with willingness-to-pay thresholds of \$50,000, \$100,000, and \$150,000 per QALY.

4.2.7 Subgroup and Sensitivity analyses

Given their significant impact on survival, we conducted prespecified subgroup analyses on witnessed OHCAs and shockable OHCAs. We also performed multiple one-way sensitivity analyses for all parameters with ranges listed in Table 4.1. To test the sensitivity to drone costs, we scaled purchase, maintenance, and per-flight costs by factors of 2 and 5, and calculated the breakeven cost multiplier at which the NMB of the drone network equals the base case. Lastly, to test sensitivity to weather and operational issues, we randomly excluded 10%, 25%, 50%, and 75% of patients from drone delivery, averaging results over 100 random draws each.

Table 4.1: Drone specifications and effectiveness and cost parameters used in analyses

Parameter	**Value**	**Range**	**Reference**
Drone specifications			Personal communication[1]
Take-off and landing duration, combined (s)	30	N/A	
Cruising speed (km/h)	56	N/A	
Purchase cost per drone base ($)	60,000	N/A	
Purchase cost per drone ($)	44,000	N/A	
Annual maintenance cost per drone ($)	5,000	N/A	
Operating cost per drone flight ($)	275	N/A	
Minimum time required to use drone AED (s)	90	[60, 120]	Starks et al. [2020]
Probability of drone AED use	0.457	[0.05, 0.75]	Bogle et al. [2019], Andersen et al. [2019], Hansen et al. [2015], Hedges et al. [2006]
Effectiveness parameters			
First-year mortality by neurological outcome at discharge			Chocron et al. [2021]
mRS 0	0.013	[0.002, 0.048]	
mRS 1	0.061	[0.031, 0.109]	
mRS 2	0.056	[0.025, 0.105]	
mRS 3	0.149	[0.093, 0.225]	
mRS 4	0.252	[0.178, 0.345]	
mRS 5	0.598	[0.442, 0.790]	
Later-year mortality per year by neurological outcome at discharge			Chocron et al. [2021]
mRS 0	0.045	[0.033, 0.059]	
mRS 1	0.023	[0.019, 0.029]	
mRS 2	0.049	[0.040, 0.061]	
mRS 3	0.047	[0.041, 0.055]	
mRS 4	0.111	[0.099, 0.123]	
mRS 5	0.096	[0.059, 0.105]	
Quality of life per year by neurological outcome at discharge			Rangaraju et al. [2017]
mRS 0	1.00	[0.85, 1.00]	
mRS 1	0.84	[0.80, 1.00]	
mRS 2	0.78	[0.60, 0.84]	
mRS 3	0.71	[0.60, 0.80]	
mRS 4	0.44	[0.34, 0.60]	
mRS 5	0.18	[0.14, 0.37]	
In-hospital costs by neurological outcome at discharge ($)			Geri et al. [2020], Dewilde et al. [2017], Geri et al. [2017]
mRS 0	54,494.27	[27247.13, 108988.54]	
mRS 1	81,686.92	[40843.46, 163373.84]	
mRS 2	128,987.95	[64493.97, 257975.90]	
mRS 3	226,696.18	[113348.09, 453392.35]	
mRS 4	281,026.97	[140513.49, 562053.95]	
mRS 5	225,442.81	[112721.40, 450885.61]	
mRS 6	22,833.10	[11416.55, 45666.19]	
First-year health system costs, by neurological outcome at discharge ($)			Geri et al. [2020], Dewilde et al. [2017]
mRS 0	3,687.31	[1843.65, 7374.62]	
mRS 1	7,822.16	[3911.08, 15644.32]	
mRS 2	14,611.32	[7305.66, 29222.64]	
mRS 3	30,197.00	[15098.50, 60394.00]	

[1] With Philip Reece, CEO of InDro Robotics (2023 June 26) and Mats Sällström, CEO of Everdrone (2023 November 14)

Parameter	**Value**	**Range**	**Reference**
mRS 4	65,906.94	[32953.47, 131813.88]	
mRS 5	74,957.60	[37478.80, 149915.19]	
Later-year health system costs per year by neurological outcome at discharge ($)			Geri et al. [2020], Dewilde et al. [2017]
mRS 0	3,830.16	[1915.08, 7660.31]	
mRS 1	5,120.29	[2560.14, 10240.57]	
mRS 2	8,523.61	[4261.81, 17047.24]	
mRS 3	22,187.18	[11093.59, 44374.36]	
mRS 4	55,509.92	[27754.96, 111019.84]	
mRS 5	52,085.63	[26042.81, 104171.26]	
Discount rate (%)	3		Assumed

4.3 Detailed Methods

4.3.1 Data Summary and Processing

Our initial data set contained $n = 25,778$ OHCAs from Ontario, Canada, from 2006-2015.

Location data processing

We first processed the locations of each OHCA. There were different types of location data which had to be processed in different ways.

From the original $n = 25778$ records, $n = 1622$ had no location data and were excluded. Of the rest with present location data, only $n = 3834$ had precise latitude/longitude, which we converted to universal transverse Mercator (UTM) coordinates. For those with truncated (2 decimal places) latitude/longitude ($n = 1110$), we concatenated random 3-digit numbers to make them as precise as the precise latitude/longitude (5 decimal places), and then converted them to UTM.

For records with truncated UTM coordinates ($n = 5321$), we concatenated random 3-digit numbers to make them full UTM coordinates. Some ($n = 27$) truncated UTM coordinates were clearly incorrect, as they had a leading digit that placed them outside the coordinate box covering Ontario (e.g., a leading digit of 9). We corrected these by replacing the leading digit with the correct one for Ontario (5 or 6 for eastings, and 4 or 5 for northings).

For those which listed the census tract ($n = 13891$), we obtained the geographic boundaries of census tracts in 2001, 2006, 2011, and 2016 from Statistics Canada [2023c], and sampled a random point within the corresponding census tract. Of these, $n = 2162$ records did not have an exact match in the list of census tracts, corresponding to 171 unique census tract IDs. We assumed that these were likely due to typographical errors. Therefore, we used a fuzzy string match, using the Jaro-Winkler distance, which is commonly used for typographical error correction [Cohen et al., 2003].

Lastly, we did a manual check of these locations by plotting them on a map of Ontario. We observed that some ($n = 235$) points were either in water or were just outside of Ontario (e.g., in Quebec). We fixed this by "rounding" all points to the nearest point within Ontario, according to the boundaries provided by Statistics Canada.

We designated OHCA locations as urban or rural based on the definitions provided by Statistics Canada. Statistics Canada uses the designation "population center" to mean a continuous area with a population of at least 1,000 with a density of at least 400 persons per square kilometer [Statistics Canada, 2023a, 2022]. All other areas are defined to be rural. We obtained the geographic boundaries of these areas [Statistics Canada, 2023b] and categorized each OHCA.

After all this, we had $n = 24,156$ points with cleaned location data.

Cleaning data

After processing locations, we removed some records with anomalous data. In particular, we removed data with negative time-to-ambulances and time-to-ambulance outliers (defined as more than 3 standard deviations above the mean, i.e., more than 1372 seconds [$\tilde{2}$3 minutes]). We also filtered out missing ages, unknown hospital states, and unknown or missing sex. We assumed that patients with missing witnessed status (n=2551) were unwitnessed. We imputed missing mRS values using a prediction model (Section 4.3.3). After all cleaning, we were left with n=22,017 records. This is summarized in Table A.1.

4.3.2 Drone Networks

We studied three different types of models: 90th percentile, coverage, and mean. For each model, we fixed a number of drones. We varied the number of drones from 5 up to 535, in increments of 5. Each drone base can have at most one drone, reflecting current logistical constraints. We also created a drone network with 538 drones, which is the maximum number of drones possible, as it is the number of candidate bases. Note that the network with 538 drones is the same regardless of objective. The drone networks were optimized according to the locations of all OHCAs for which we had valid location data ($n = 24,156$).

The 90th percentile models minimize the average of the upper decile of the response time with a given number of drones. Technically, these models are referred to as conditional value-at-risk models (CVaR) [Boutilier and Chan, 2022]. There were 107 90th percentile models.

The coverage models maximize how many OHCAs can be reached in a given amount of time with a given number of drones. We ran coverage models for coverage times of 1 to 7 minutes, in 1 minute increments. This yielded a total of 749 coverage models.

Lastly, the mean models minimize the mean response time with a given number of drones. Technically, these are referred to as p-median models [Boutilier and Chan, 2022]. There were 107 median models.

Including the unique network with 538 drones, we evaluated 964 drone networks.

The possible locations that a drone can be placed were current existing EMS, fire, and police stations in Ontario. The time that it takes a drone placed on a base to reach a given OHCA is calculated by first finding the distance between the base and the OHCA. Then, we assume 30 seconds for takeoff and landing, and a speed of 56 km/h throughout the flight, to calculate the time.

All models were solved using Gurobi and Python, on an Intel i7 processor, with a time limit of 10 hours. For the CVaR models, we set an optimality gap of 1% for tractability reasons.

4.3.3 Prediction Models

Training models

We built three predictive models using our dataset, detailed in Table A.4.

To train each model, independent 80% subsets were drawn to fit the model, with the remaining 20% used for validation. For the mRS model, we first filtered to those who were admitted to the hospital. An XGBoost model [Chen and Guestrin, 2016] was used for the mRS model, as we found it performed much better than an ordered multinomial logistic regression model. We performed hyperparameter tuning for the XGBoost model using a grid search, with 5-fold cross-validation (parameters listed in Table A.4).

The shockable model achieved an area under the receiver operator characteristic curve (AUC) of 0.63 and the survival to hospital model achieved an AUC of 0.74. The XGBoost model achieved a multiclass AUC [Hand and Till, 2001] of 0.57.

Applying the models

When we apply the models in the drone setting, we substitute the drone response time (if the drone is used) to calculate the new outcome. If the drone is not used, then we do not apply the prediction models and simply revert to the historical outcome from the data set.

We first predict the new shockable likelihood. We then use this as an input into the other two models as follows. Let the predicted probability of a shockable rhythm be P_{shock}, the survival model be $s()$, the mRS model $m()$, and let the patient's covariates, not including shockability, be represented by the vector x. Then, the predicted survival is calculated as:

$$P_{\text{shock}} \cdot s(x, \text{shock} = 1) + [1 - P_{\text{shock}}] \cdot s(x, \text{shock} = 0)$$

And the predicted mRS is calculated as:

$$P_{\text{shock}} \cdot m(x, \text{shock} = 1) + [1 - P_{\text{shock}}] \cdot m(x, \text{shock} = 0)$$

Note that all three models output probabilities: e.g., the mRS model outputs a list of probabilities over mRS 0 through 6, for each patient. We propagate these probabilities throughout our calculations in order to calculate the expected values for all measures.

Enforcing monotonicity of predictions

The prediction models introduce a statistical artifact which has to be corrected. This can be seen through the following example. Say an ambulance takes 3 minutes to reach a patient, and upon arriving the patient has a shockable rhythm. Now, say a drone takes 2 minutes to reach a patient, and using this 2 minute value we predict a new shockable likelihood. However, the prediction model will almost surely assign a non-zero probability to the not shockable outcome. This paradoxically leads to a lower expected utility for the patient, despite the drone arriving earlier.

This can be corrected by enforcing a constraint that a drone arriving earlier should not lead to a worse outcome than the ambulance outcome - in other words, outcomes should be monotonic with respect to response time. For the shockable model, this means that if a drone arrives earlier, and the patient was shockable in the ambulance case, then they will definitely (i.e., with 100% probability) be shockable when the drone arrives. Similarly, for the survival model, if the patient survives in the ambulance case, then if the drone arrives earlier they will definitely survive as well.

For the mRS model, we can enforce this constraint by saying that the patient can never go to a worse mRS status than the ambulance outcome, if the drone arrives earlier. We do this as follows:

1. Get the mRS prediction as a list of probabilities over each mRS state.
2. Set the probabilities of mRS' worse than the ambulance case to 0.
3. Normalize the values by dividing by the sum, to make them valid probabilities again.

Model for imputing missing mRS

In the dataset, $n = 1006$ patients survived hospital discharge but had a missing mRS. First, we analyzed their missingness by comparing important covariates between those who survived discharge and had mRS, and those who survived but had missing mRS. We found statistically significant differences in some covariates (see Table A.5). We also used Little's test to test if data was missing completely at random, which returned $p < 0.001$, indicating we can reject the null hypothesis that the data was missing completely at random. As a sensitivity analysis, we also excluded those with missing mRS (Figure A.11) and found all drone networks were still cost-effective.

Therefore, we imputed their standard care mRS using a prediction model. We used an XGBoost model similar to the one used above for the analysis. First, we filtered for patients who survived hospital discharge (i.e., mRS 0-5). Then, we drew a random 80% training set, on which we performed 5-fold cross validation for hyperparameter tuning. The covariates used were: response time, age, sex, witnessed, and shockability. The following hyperparameters were chosen based on hyperparameter tuning:

- Max depth: 3
- Eta: 0.1
- Gamma: 0.3
- Number of rounds: 50
- Column sample by tree: 0.8
- Min child weight: 0.8
- Subsample: 0.8
- Number of threads: 2

The model achieved a multiclass AUC of 0.51 on the test set. We then applied this model to the records with missing mRS, and took the class with the highest predicted probability. The predicted classes for the $n = 1006$ patients are in Table A.6.

4.3.4 Cost-Effectiveness Model Parameters

Drone costs

These values were obtained via personal communications with Philip Reece, CEO of Indro Robotics and Mats Sällström, CEO of Everdrone, in 2023. The cost of the drone network has three components. Firstly, each drone must be housed in a drone port. A drone port costs \$30,000, and lasts for 5 years. In our simulations, we run a drone network for 10 years, so this amounts to a cost of \$60,000 per drone. Secondly, each drone costs \$35,000; additionally, the camera costs \$8,000, and the payload \$1,000. This yields a total cost of \$44,000/drone to purchase. Third, each drone costs \$5,000/year for maintenance. Lastly, every time the drone flies, it costs \$250 in operator costs; and, every 50 launches, the drone port has to be inspected, at a cost of \$1,250. This means a cost of \$2.75 per drone launch. We assume that a drone is sent for every OHCA.

Overall, the cost of a drone network with n_{drones} drones and $d_{\text{dispatches}}$ dispatches is:

$$154000 \cdot n_{\text{drones}} + 275 \cdot d_{\text{dispatches}}$$

Note the first term above captures the fixed cost of the drone network, and the second term the variable cost.

Transition probabilities

The transition probabilities are the yearly mortality rates for each discharge mRS. We distinguish between first-year and later year. Chocron et al. [2021] report both 1-year and 5-year mortality rates of OHCA patients in King County, Washington. We take the 1-year rates directly, and impute the later-year mortalities with the 5-year mortalities. To calculate lower and upper bounds, we simply repeat this procedure for the lower and upper bounds of the 95% confidence intervals that they report.

Health care costs

For costs, the most relevant paper in the Ontario context is Geri et al. [2020]. However it only provides the average costs of an OHCA in Ontario. In order to separate it out by mRS, we use cost data from other papers, and make the assumption that the between-mRS ratios are constant. For example, if a mRS2 patient is twice as costly as an mRS1 patient in New York, it would also be twice as costly in Ontario. Based on this formulation, we can impute per-mRS costs for in-hospital costs by solving a system of equations.

Mathematically, we know the average cost $\bar{c}$ from the literature. We also know from our data set that we have n_i patients with an mRS of i (i.e., ranges from 0 to 6). We know that the total number of patients is n. We want to calculate c_i, i.e., the costs for each mRS. So, by definition, we have the weighted average:

$$\frac{1}{n}\sum_{i=0}^{6} n_i \cdot c_i = \bar{c}$$

Now, we introduce the between-ratios for each mRS, relative to the cost of mRS 0. We let x_i be the ratio of costs from mRS i to mRS 0. If we take each of these values from the literature, we can solve for each c_i via standard substitution.

Using cost data from Dewilde et al. [2017] on stroke patients, we calculate the ratios of treatment cost for all mRS' relative to mRS 0. However, Dewilde et al. [2017] does not provide data on mRS 6 patients. To calculate that ratio, we use data from Geri et al. [2017] which provides OHCA costs by cerebral performance category (CPC). We use the ratio between CPC 5 (death, equivalent to mRS 6) and CPC 1 (the least severe CPC category) as the ratio between mRS 6 and mRS 0. Given these ratios, and knowing the number of people in each mRS category in our data set, we can solve this system of equations and calculate the cost for each mRS.

We follow a similar approach for post-hospital costs, and separate into first-year and later-year. Dewilde et al. [2017] gives 3 month, first-year, and later-year costs, while Geri et al. [2017] gives 1 month costs. First, based on the ratios between the 3-month costs from Dewilde et al. [2017], we calculate the by-mRS 1-month costs, as we did above. Then, from Dewilde et al. [2017], we calculate the ratio of first-year costs to 1-month costs, and later-year costs to 1-month costs. From these ratios, we scale up the per-mRS costs that we previously calculated.

In this way, all cost values are imputations based on Geri et al. [2020], which provides costs in terms of 2015 Canadian dollars. We use the Consumer Price Index (CPI) from the Bank of Canada to calculate the equivalent costs in 2023 dollars. The CPI increased 23.72% from 2015 to 2023.

For the lower and upper bounds, we simply set the lower bound as half of the value, and the upper bound as twice it.

Utilities

The utilities are simply taken from Rangaraju et al. [2017], and in this case we do not separate between the first-year and later-year. The upper and lower bounds are taken to be the 25th percentile and the 75th percentile as reported in that paper.

Probability of drone use

This value was taken from Bogle et al. [2019], who themselves take this value from Hansen et al. [2015]. This value is not specific to drones: it is instead the probability of a trained bystander performing CPR. However, it is in line with the value used by Andersen et al. [2019] (a cost-effectiveness analysis of public AEDs) of 47%, which is taken from Hedges et al. [2006]. As this value is poorly studied in the literature, we chose a large range of 0.05 to 0.75 for the possible values of this probability in our sensitivity analysis.

Minimum time to use drone

This value was taken from Starks et al. [2020], which is a simulation study. We assumed a range of 60 seconds to 120 seconds for the sensitivity analysis.

Running the Cost-Effectiveness Model

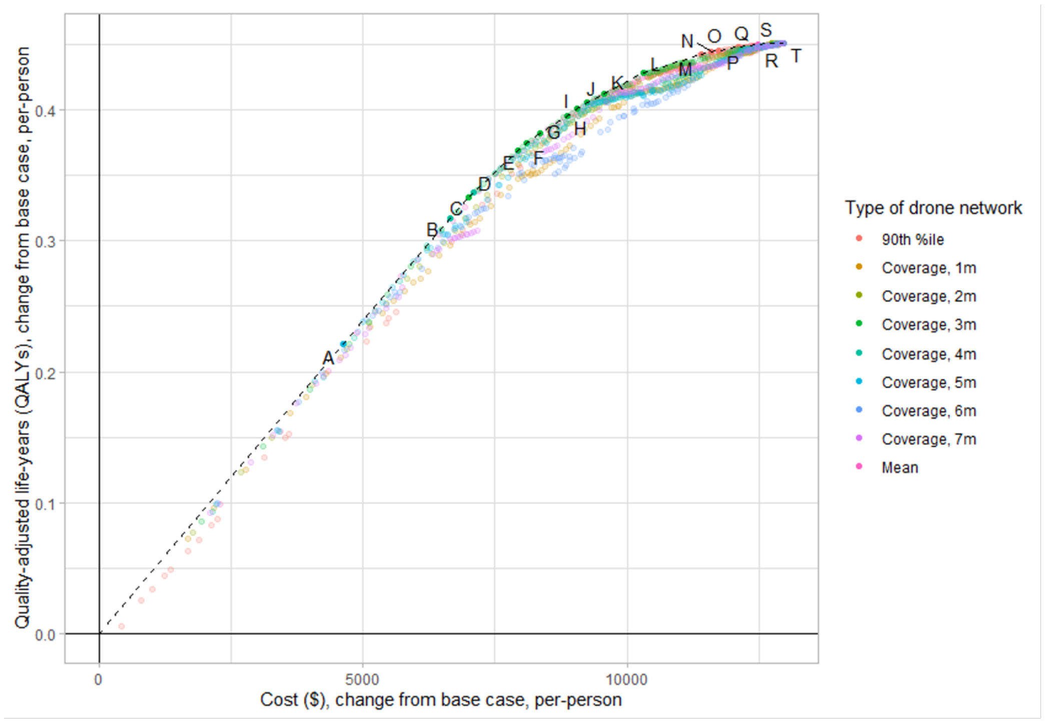
First, for a given drone network, we calculate the time it would take to the nearest drone. If the difference between the drone time and the ambulance time is less than the minimum time to use, then we revert to standard care. Otherwise, the AED is used with some probability (and if not used, again revert to standard care). As described above, we predict counterfactual outcomes based on the drone time.

We run the Markov models for the rest of the life of the patient. Technically, we use the formula for the value of an infinite-horizon discounted Markov reward process, from Chaper 1.5, to calculate the total discounted cost and total discounted utility for each mRS discharge state. Then, knowing the probability of the patient being in each mRS state, given either by the data (for standard care), or by the prediction models (for the drone case), we can calculate the expected cost and expected utility.

We obtain the initial distribution vector, $\boldsymbol{\pi}$, either from our prediction models (recall they output a probability vector over mRS states) or by historical data.

Note there is a caveat in the term discount factor as used in Markov models versus how it is typically used in health economics. In economics, typically the present value of a reward R in time period t is given by $R/(1+\lambda)^t$, while in Markov processes it is $R \cdot \lambda^t$. So, a discount factor of 3%, as is typical in cost-effectiveness analysis, cannot be plugged directly into this formula (in fact it would extremely heavily favor the present). We make the appropriate modifications before using this formula, i.e., using $\lambda = 1/(1+0.03) = 0.97$.

Figure 4.2: Cost-QALY plane. Labeled points (and dotted line) indicate the efficient frontier; labels correspond to Table 2



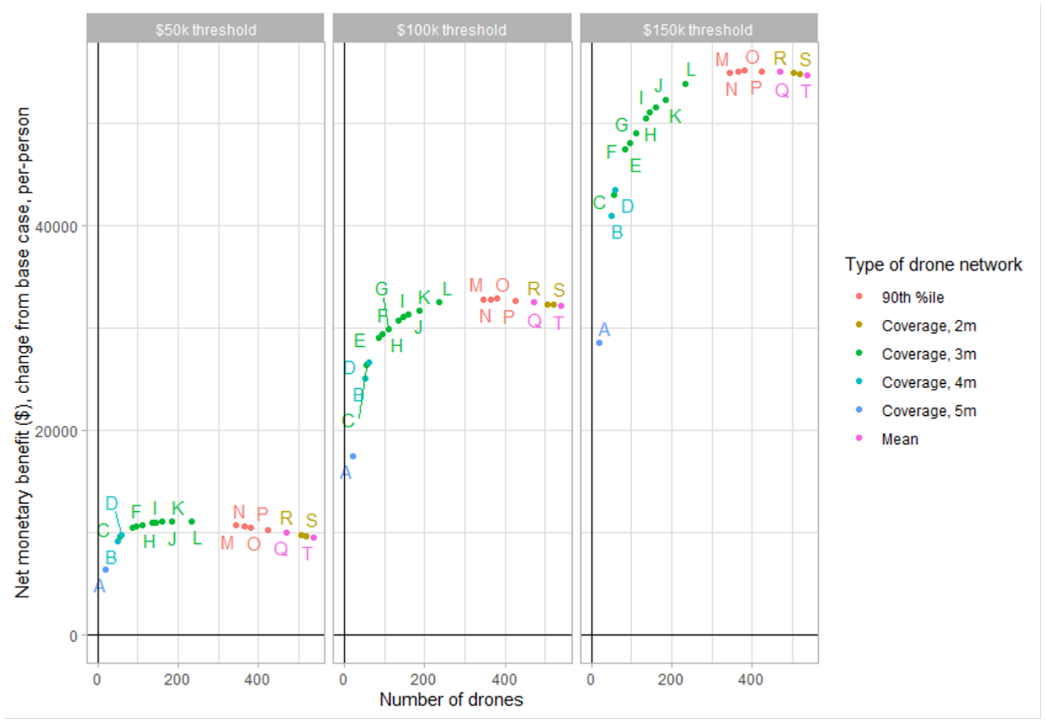
4.4 Results

A total of 25,778 EMS-treated OHCAs were recorded during the study period. After applying the exclusion criteria, 24,156 were used to optimize the drone networks, while 22,017 OHCAs were used for outcomes analysis as they had all relevant patient covariates (Figure A.1, Table A.1). The final cohort had a mean age of 68.6 years (SD 16.9), was 63.7% male, and 93.8% lived in urban areas.

All drone networks had higher costs and higher QALYs than the base case. After applying strict and extended dominance, 20 drone networks were on the cost-QALY efficient frontier (Figure 4.2), which we refer to as the "non-dominated networks". Fourteen of the 20 networks were optimized for coverage, one optimized for mean response time, four optimized the 90th percentile, and the remaining one was the unique 538 drones network. At an ICER threshold of $50,000 per QALY gained, the network with the highest QALYs was the 3-minute coverage network with 235 drones; at thresholds of $100,000 and $150,000, it was the 90th percentile network with 380 drones.

Table 4.2 displays characteristics of the response time distribution, survival outcomes, and cost-effectiveness for the base case and each of the non-dominated networks. In the base case, mean response time was 6 min 21 s. There were 11,691, 1,855, and 1,748 survivors to hospital admission, discharge, and with neurologically favorable outcome, with respective costs per survivor of $41,864.15, $263,845.69, and $279,996.42. Each of the 20 drone networks had shorter response times, more survivors across all categories, and higher costs per survivor than the base case. Average drone network response times ranged from 4 min 18 s to 1 min 52 s, representing 32% to 71% reductions from the base case. The expected number of survivors to admission, discharge, and with neurologically favorable outcome were in the range [12,909, 13,915] (+10% to +19%), [2,263, 2,704]

Figure 4.3: Net monetary benefit, at thresholds of $50k, $100k, and $150k, for frontier networks. Point labels correspond to Table 2



(+21% to +46%), and [2,097, 2,448] (+20% to +40%).

Figure 4.3 shows the net monetary benefit (NMB) for the 20 networks at the three thresholds of $50,000/QALY, $100,000/QALY, and $150,000/QALY. For every threshold, all networks had a higher NMB than the base case. As networks get larger, the NMB plateaus and then slightly declines. This trend is also seen when looking at all 964 networks in Figure A.2, showing that the NMB plateaus at around 100 drones. For smaller networks (¡100 drones), those with the highest NMB tended to be the mean, 4-minute coverage, and 3-minute coverage networks (Table A.2). 90th percentile networks typically had the highest NMB for large networks (300-500 drones), while 2-minute coverage networks were best for the largest networks.

4.4.1 Subgroup Analyses

In Table A.3, we summarize clinical outcomes for the four subgroups of interest in the base case and for each of the 20 non-dominated networks. For witnessed OHCAs, survival to hospital discharge was 12.8% in the base case and [16.0%, 19.2%] over the 20 drone networks, representing relative increases of 25% and 50%. For shockable OHCAs, survival to hospital discharge was 27.7% in the base case and [31.5%, 35.7%] over the 20 networks, representing relative increases of 14% and 29%. For shockable and witnessed OHCAs, survival to hospital discharge was 28.8% in the base case and [33.3%, 37.9%] over the 20 networks, representing relative increases of 16% and 32%. Compared to the general population, patients with shockable rhythms or witnessed arrests had higher base case NMBs, and experienced greater gains in their NMBs with drone networks (Figure A.3). Patients that were originally non-shockable also experienced some benefit due to the increased probability of being shockable if the drone arrives earlier than historical ambulance response.

4.4.2 Sensitivity Analysis

Sensitivity of clinical outcomes to changes in model parameters for the 20 non-dominated networks is shown in Figures A.4, A.5, and A.6. Clinical outcomes were most sensitive to the probability of drone AED use, but across all parameter ranges tested, all 20 networks had more survivors across all categories than the base case.

Sensitivity of NMB to model parameters is shown in Figure A.7. The NMB was generally most sensitive to the probability of drone AED use, later-year mortality for mRS 0, utility for mRS 0, minimum time to use drone, and in-hospital costs for mRS 0. Larger networks, which have higher NMBs, were also more sensitive to parameter values. All 20 networks maintained a higher NMB than the base case across all parameter ranges tested.

When drone operational costs are increased two-fold or five-fold, many of the 964 drone networks still have a higher NMB than base case at a $50,000 threshold, and all but one remained so at the higher thresholds (Figure A.8). Cost-effectiveness is highly robust to increases in drone costs: at a $50,000 threshold, drone costs can increase by nearly ten-fold and networks with fewer than 100 drones still have a higher NMB than the base case (Figure A.9).

Lastly, while the NMB decreases as weather conditions exclude patients from drone delivery, all networks remained cost-effective (Figure A.10), even when 75% of patients are excluded.

Table 4.2: Response time distribution metrics, outcome metrics, and cost-effectiveness metrics for the base case (i.e., historical response) and each drone network configuration that has a non-dominated incremental cost-effectiveness ratio (ICER). Point labels correspond to Figure 2. Response time for a network is calculated as the minimum of the ambulance time and the drone time.

Point	**Drone network**	**Response time (s)**		**Expected number of patients, count (%)**			**Total cost ($)**					**QALYs**	**NMB ($, millions) at threshold**			**ICER ($/QALY)**
		Mean (SD)	*Median (IQR)*	*Survived to hospital admission*	*Survived hospital discharge*	*Discharged with neurologically favorable status (mRS 0-2)*	*Total (in millions)*	*Per-person (n=22,017)*	*Per survivor to hospital admission*	*Per survivor to hospital discharge*	*Per neurologically favorable survivor*		*$50,000*	*$100,000*	*$150,000*	
	Base case	381.26 (151.83)	358.00 (292.00 - 433.00)	11691.00 (53.10)	1855.00 (8.43)	1748.00 (7.94)	489.43	22229.81	41864.15	263845.69	279996.42	24328.61	727.00	1943.43	3159.86	N/A
A	Coverage, 5min, 20 drones	258.00 (143.65)	234.37 (174.69 - 300.00)	12909.02 (58.63)	2263.05 (10.28)	2097.33 (9.53)	591.24	26853.81	45800.55	261257.93	281900.80	29197.03	868.61	2328.46	3788.31	20911.61
B	Coverage, 4min, 50 drones	207.73 (137.86)	180.00 (133.61 - 228.04)	13378.25 (60.76)	2442.65 (11.09)	2247.90 (10.21)	635.72	28873.93	47518.73	260257.28	282805.10	31309.88	929.78	2495.27	4060.77	21050.67
C	Coverage, 3min, 55 drones	197.10 (146.69)	157.82 (115.83 - 219.82)	13439.18 (61.04)	2473.10 (11.23)	2271.93 (10.32)	643.73	29237.68	47899.20	260291.20	283339.39	31664.13	939.48	2522.69	4105.89	22607.52
D	Coverage, 4min, 60 drones	197.42 (138.44)	170.11 (122.77 - 215.12)	13449.98 (61.09)	2480.28 (11.27)	2278.58 (10.35)	645.74	29329.38	48010.85	260351.66	283398.11	31743.54	941.43	2528.61	4115.79	25425.12
E	Coverage, 3min, 85 drones	176.34 (139.21)	141.37 (105.44 - 181.31)	13594.78 (61.75)	2542.09 (11.55)	2327.52 (10.57)	664.30	30172.19	48864.42	261321.23	285411.53	32458.95	958.65	2581.59	4204.54	25937.77
F	Coverage, 3min, 95 drones	173.16 (137.65)	138.92 (103.59 - 178.90)	13612.57 (61.83)	2551.71 (11.59)	2335.66 (10.61)	667.76	30329.32	49054.73	261691.69	285897.59	32573.11	960.89	2589.55	4218.21	30305.71
G	Coverage, 3min, 110 drones	168.16 (134.38)	135.77 (102.14 - 173.16)	13641.36 (61.96)	2565.95 (11.65)	2346.74 (10.66)	673.38	30584.66	49363.30	262429.70	286943.80	32737.73	963.50	2600.39	4237.28	34150.32
H	Coverage, 3min, 135 drones	159.07 (126.34)	130.32 (97.23 - 165.55)	13704.59 (62.25)	2593.68 (11.78)	2367.11 (10.75)	684.46	31087.61	49943.55	263893.96	289152.02	33039.67	967.53	2619.51	4271.49	36673.89
I	Coverage, 3min, 145 drones	155.49 (125.65)	126.75 (94.82 - 162.11)	13723.92 (62.33)	2603.98 (11.83)	2374.22 (10.78)	688.71	31280.88	50183.25	264484.49	290079.47	33150.62	968.82	2626.35	4283.88	38350.29

Point	Drone network	Response time (s)		Expected number of patients, count (%)			Total cost ($)					QALYs	NMB ($, millions) at threshold			ICER ($/QALY)
		Mean (SD)	*Median (IQR)*	*Survived to hospital admission*	*Survived hospital discharge*	*Discharged with neurologically favorable status (mRS 0-2)*	*Total (in millions)*	*Per-person (n=22, 017)*	*Per survivor to hospital admission*	*Per survivor to hospital discharge*	*Per neurologically favorable survivor*		*$50,000*	*$100,000*	*$150,000*	
J	Coverage, 3min, 160 drones	152.57 (123.36)	124.55 (94.14 - 158.13)	13739.15 (62.40)	2612.59 (11.87)	2381.64 (10.82)	692.93	31472.50	50434.71	265227.03	290947.03	33253.57	969.75	2632.43	4295.10	40984.58
K	Coverage, 3min, 185 drones	147.62 (112.35)	124.01 (93.92 - 156.93)	13767.61 (62.53)	2626.02 (11.93)	2392.08 (10.86)	699.97	31792.26	50841.81	266551.73	292619.25	33404.93	970.28	2640.52	4310.77	46511.58
L	Coverage, 3min, 235 drones	133.12 (98.25)	114.69 (85.57 - 146.44)	13839.45 (62.86)	2657.84 (12.07)	2414.70 (10.97)	716.53	32544.59	51774.76	269592.76	296738.64	33747.72	970.85	2658.24	4345.62	48321.19
M	90th %ile, 345 drones	120.46 (89.29)	101.78 (75.73 - 134.09)	13884.92 (63.06)	2686.04 (12.20)	2434.60 (11.06)	740.89	33650.74	53359.21	275829.23	304316.55	34061.72	962.20	2665.28	4368.37	77559.40
N	90th %ile, 365 drones	118.55 (85.02)	100.81 (75.39 - 132.99)	13890.33 (63.09)	2690.00 (12.22)	2437.37 (11.07)	744.96	33835.59	53631.42	276936.54	305640.73	34106.02	960.34	2665.64	4370.94	91887.67
O	90th %ile, 380 drones	117.33 (83.25)	100.27 (75.00 - 132.05)	13895.67 (63.11)	2692.67 (12.23)	2439.68 (11.08)	747.86	33967.42	53819.70	277739.69	306541.01	34137.07	958.99	2665.85	4372.70	93457.77
P	90th %ile, 425 drones	115.24 (79.61)	99.42 (74.24 - 131.09)	13904.47 (63.15)	2697.69 (12.25)	2443.17 (11.10)	756.08	34340.68	54376.66	280268.64	309466.29	34189.48	953.40	2662.87	4372.34	156807.56
Q	Mean, 470 drones	113.39 (75.37)	97.72 (72.50 - 130.12)	13911.17 (63.18)	2701.25 (12.27)	2445.58 (11.11)	764.04	34702.40	54922.98	282848.26	312418.14	34227.84	947.35	2658.74	4370.13	207641.98
R	Coverage, 2min, 505 drones	112.77 (73.24)	97.69 (72.49 - 129.88)	13912.77 (63.19)	2702.68 (12.28)	2446.68 (11.11)	769.69	34958.98	55322.69	284788.32	314586.71	34245.36	942.58	2654.84	4367.11	322400.68
S	Coverage, 2min, 520 drones	112.39 (72.72)	97.56 (72.46 - 129.69)	13914.29 (63.20)	2703.36 (12.28)	2447.15 (11.11)	772.19	35072.52	55496.29	285641.62	315546.90	34252.52	940.43	2653.06	4365.69	348832.27
T	538 drones	112.11 (72.41)	97.40 (72.22 - 129.57)	13914.55 (63.20)	2704.01 (12.28)	2447.60 (11.12)	775.12	35205.55	55705.78	286656.13	316685.68	34259.54	937.86	2650.83	4363.81	417242.35

4.5 Discussion of Main Results

This study is the most comprehensive cost-effectiveness analysis to date of drone networks for AED delivery for OHCA response. All 964 networks were cost-effective regardless of the willingness-to-pay threshold. We identified 20 drone networks on the cost-QALY efficient frontier, all of which had higher QALYs, better patient outcomes, and shorter response times than the base case (i.e., no drones). Witnessed or shockable patients experienced the largest improvements in clinical outcomes and NMB. However, even non-shockable patients (in the base case) may benefit due to becoming shockable with an earlier drone response, leading to a potential benefit for them as well (Table A.3). Our sensitivity analyses showed that even for the most pessimistic parameter values, all 20 drone networks remained cost-effective and had better clinical outcomes than the base case.

NMB increased as more drones were added to the network, plateauing around 100 drones. Given our study region's population and size, this translates to having roughly 1 drone per 70,000 people or 1 drone per 260 square km to maximize the NMB. The largest drone network corresponded to 1 drone per 13,000 people or 1 drone per 48 square km, implying that drone-delivered AEDs will remain cost-effective for large networks. Recent studies in Sweden piloted networks of three to five drones [Schierbeck et al., 2022a, 2023]. Relative to the population and region size, these represent roughly 1 drone per 27,000 to 40,000 people or 1 drone per 39 to 42 square km, which is within or near the range of drone network sizes that we show to be cost-effective. While real-world drone networks to date have been small, our findings suggest that they will be cost-effective as they scale up in the future.

In the literature on optimizing emergency response using ambulances, drones, or static AEDs, many types of models have been considered, optimizing for coverage, mean response time and upper percentile response times [Boutilier and Chan, 2022, Sun et al., 2018, Karlsson et al., 2019]. In this paper, we found that networks optimized for coverage tended to be the most cost-effective at each willingness-to-pay threshold. Indeed, 14 of the 20 non-dominated networks optimized for coverage. For networks with fewer than 100 drones, optimizing for 3 or 4-minute coverage tended to have the highest NMB. Given how quickly survival declines as a function of time, it is intuitive that optimizing coverage, equivalent to setting a maximum response time, would make the best use of limited resources.

The three studies closest to this paper are the economic analyses conducted by Bogle et al. [2019], Bauer et al. [2021], and Röper et al. [2023]. They all found that drones were cost effective, but under restrictive assumptions. None used patient-level OHCA data, relying instead on estimated incidence from population data. They also did not consider healthcare costs, which comprise most of the system costs. The first two studies used fixed probabilities of survival as a function of response time interval, without considering the effect of patient covariates, while the third focused on comparing coverage of drones against static AEDs, without considering survival outcomes. In contrast, we used real OHCA data allowing us to generate machine learning-based outcome predictions, adjusting for relevant covariates, to properly estimate the counterfactual of improved response time. We also included healthcare costs that were stratified by neurological outcome, providing significant granularity in our calculations. Andersen et al. [2019] also studied cost-effectiveness of static AEDs using a decision tree and Markov model but measured neurological outcome as cerebral performance category (CPC), estimated using coarse probabilities based only on response time intervals. A recent Swedish government report also reported cost-effectiveness of a five-drone network [Fledsberg and

Pauli, 2023].

While our analysis focused on drone delivery of AEDs for OHCA, there is increasing interest in drone delivery of other medical supplies such as naloxone kits for opioid overdose, epinephrine autoinjectors for anaphylactic shock, and trauma kits. Drone networks that deliver these other items in addition to AEDs will likely have improved cost-effectiveness as the drone costs will be spread over more deliveries.

Drones are a promising technology that can decrease response times, meaning more patients can receive treatment within a clinically meaningful time. However, success depends critically on other factors like bystander availability who can retrieve and apply the AED. Overall, for the benefits of drone delivery to be realized, both rapid delivery and effective bystander integration are needed.

Limitations Our study has several limitations. First, two key parameters for our model – the probability of drone AED use and the minimum time to use the drone AED – are not well-studied in the literature. Therefore, we conservatively assumed large ranges for these parameters. For probability of AED use we considered a range of 0.05 to 0.75; even for the smallest value, our drone networks were still cost-effective and associated with better clinical outcomes. Further research on human-drone interaction [Unity Health Toronto, 2023, Sasson et al., 2010] may help to refine these estimates. Second, variables like socio-economic factors or co-morbidities were not available in our dataset, which may limit our predictive models.

As described in Section 4.3.4, healthcare costs and utilities stratified by mRS were only available for stroke patients. To address this issue, we employed an imputation technique to estimate the analogous values for OHCA. These values may be affected by region-specific differences in treatments. However, our sensitivity analysis showed that our findings were robust to uncertainties in these parameters. We also imputed missing mRS values for survivors (Section 4.3.3), but our findings were robust to imputation (Figure A.11).

Drone operations are affected by drone technology, weather, restricted airspace, and dense urban environments, which were not explicitly modeled. However, the drone networks remained cost-effective even when up to 75% of patients, as seen in real-world implementations [Schierbeck et al., 2023], are not dispatched a drone. There may also be unforeseen implementation costs, but we found robustness to very large increases in costs.

4.6 Re-Analysis with our Algorithms

4.6.1 Sensitivity Analysis

Now, we perform a much more thorough sensitivity analysis using the tools developed in Chapter 2. We re-implemented the decision tree and Markov model from this chapter using `markovag`, our software from Chapter 2. As an example, we study the 20-drone network that maximized 5-minute coverage of cardiac arrests, as this was the smallest cost-effective network. We performed a three-way sensitivity analysis on the following three key parameters:

- m: the drone cost multiplier. Maaz et al. [2024] obtained estimates of the cost to operate the drone network from two drone companies. However, as the technology evolves and the complexity of operating a large network changes (currently, the largest operating network in the world, in Sweden, has only five drones), this cost may also change in unexpected ways. Thus, the estimated drone costs can be scaled by m. In the original paper, values of $m \in \{2, 5\}$ were tested.

- p_u: the probability of the drone-delivered AED being used. A value of 0.457 was used for this probability, obtained from a prior study on bystander use of static AEDs. However, there remains significant uncertainty about the true usage probability and how it might vary across geographies. Thus, in the original one-way sensitivity analysis, a range of $[0.05, 0.75]$ was tested.

- p_0: the first-year mortality of a patient discharged with mRS 0. In the original paper, this parameter was found to be one of the most influential on NMB. It had a default value of 0.013, and in the original one-way sensitivity analysis, a range of $[0.002, 0.048]$ was tested.

We used `markovag.markov` to symbolically calculate the difference in NMB between the drone case and the no-drones case. Then we used `markovag.cad` to construct the CAD for the inequality representing "drone NMB $\geq$ status quo NMB". The most important cell in the CAD is

$$0 < p_u < 1 \quad \wedge \quad 0 < p_0 < 1 \quad \wedge \quad 0 < m \leq -132.46 p_0 p_u + 154.46 p_u$$

We omit the other cells as they represent extreme cases that are uninteresting or unrealistic for the policymaker (e.g., $p_u = 1$, meaning universal drone AED use). Some insights can be gleaned by analyzing the first derivatives of the multilinear function bounding m:

- With respect to p_u: we have $-132.46p_0 + 154.46$, which is strictly positive for $0 < p_0 < 1$, so the bound on m is strictly increasing in p_u. As expected, with a higher probability of drone AED use, higher drone costs are acceptable due to the larger benefits.

- With respect to p_0: we have $-132.46p_u$ which is strictly negative for $0 < p_u < 1$, so the bound on m is strictly decreasing in p_0. This result is not easy to obtain otherwise and is somewhat unexpected. A higher probability of death leads to lower utilities obviously, but also lower costs, as there are non-trivial costs associated with treating surviving patients. In this case, the analysis reveals that the increased costs outweigh the increased benefits, so that with a higher probability of death, the acceptable level of drone costs *decreases*.

From this CAD representation, a policymaker can trace a path down the tree to determine the validity of a set of parameters.

For illustrative purposes, we visualize the parameter regime over which the drone network is cost-effective, found by `markovag`, by fixing $m \in \{1, 5\}$ and letting p_0 and p_u vary. For comparison, we consider the usual one-way and two-way sensitivity analyses using a grid with five evenly spaced points, $\{0, 0.25, 0.5, 0.75, 1\}$. For the one-way analysis, we fixed one parameter at the default value according to Table 4.1 ($p_0 = 0.013, p_u = 0.457$) and tested the free parameter at the five points above. For the two-way sensitivity analysis, we construct a 5x5 grid of the two parameters. Grid points where the inequality holds (does not hold) are denoted by a green star (red cross).

The plots in Figure 4.4 show that the range of parameter values associated with cost-effectiveness is quite large, much more than suggested by the one-way or even two-way analyses, and that the nonlinear boundary is easily identified. Methodologically, a grid search will always miss some part

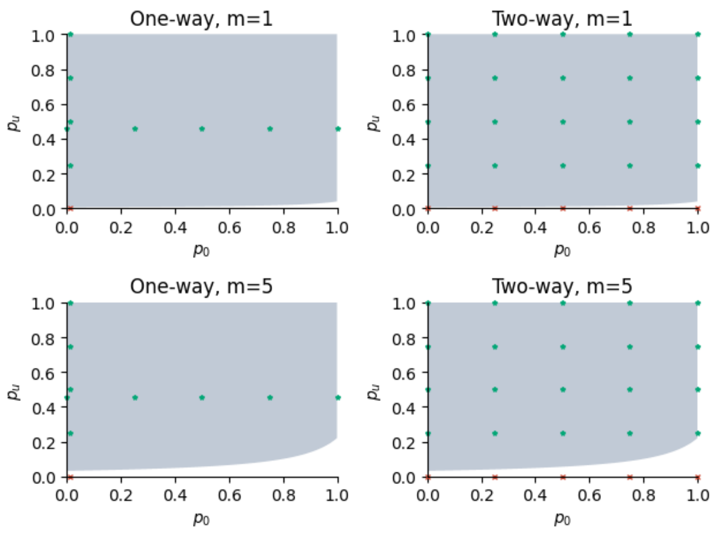


Figure 4.4: Visualization of the parameter space over which the drone network is cost-effective. The shaded gray region is the exact analytic solution obtained by `markovag`. The points represent the traditional mesh grid approach, with green representing a valid point and red invalid.

of the true cost-effective region and it remains difficult to know a priori how fine a grid is needed to approximate well the boundary. In the context of the drone application, these results elucidate that the probability of drone AED usage may be much lower than expected while still leading to a cost-effective intervention. Also, sensitivity of the cost-effectiveness finding to p_0 seems to be more apparent only when considering the interaction with other parameters.

4.6.2 Subgroup Analysis

We can also perform a much more rigorous subgroup analysis using the tools from Chapter 3. Recall that in this chapter's setup, the patient's discharge mRS is given by an ML model (see Figure 4.1) that takes as input the patient's age and sex, features of the OHCA, and the time-to-defibrillation. This means that $\boldsymbol{\pi}$ is linked to the outputs of an ML model. We constructed the Markov model from Figure 4.1 in our software package from Chapter 3, `markovml` and embedded a pretrained classifier[2]from the patient data that we used in this chapter.

By setting appropriate constraints to build $\mathcal{X}$, in the language of Chapter 3, we can now perform subgroup analyses. For example, we found that the best-case (or, maximum) NMB for men is about 50% higher than for women. This reflects lower mortality among men, i.e., lower probability of being discharged with mRS 6, which would cause them to accrue zero utility and hence zero NMB. Indeed, this finding is corroborated by other findings in the healthcare literature which find that women have at least 50% higher mortality than men after an OHCA Parikh et al. [2020]. Notably, doing this analysis as an optimization problem allowed us to solve it in an automated way, without requiring the underlying patient data nor simulating representative patients.

[2]In this chapter's analysis, a multiclass XGBoost classifier is used. Due to limited support for multiclass classification models in `markovml` at time of writing, we opted for a binary decision tree for this re-analysis, that predicted the probability of a patient being discharged with mRS 6 versus another mRS.

4.7 Conclusion

In this cost-effectiveness analysis, we found that drone-delivery of defibrillators to individuals experiencing OHCA is a cost-effective intervention that improves response times and survival. These results are robust to a variety of modeling assumptions. We then performed re-analyses of drone-delivered defibrillators using the Python packages developed in this thesis, namely `markovag` and `markovml`. The exact multi-way sensitivity analysis enabled by `markovag` shows us that drones are even more cost-effective than our initial analysis led us to believe, and the exact subgroup analysis enabled by `markovml` rigorously allowed us to compare male and female patients, and this analysis can be easily extended to more complex subgroups.

Chapter 5

Conclusion

In this thesis, we set out to study the analysis of Markov processes whose parameters are not fixed, i.e., are *uncertain*. With that motivation, this thesis made two major technical contributions and one major empirical contribution:

- Chapter 2 showed how to reduce sensitivity analysis to an equivalent polynomial system, and, exploiting the properties of this polynomial system, we introduced a variant of CAD that has only singly exponential time complexity.
- Chapter 3 introduced the problem of verifying Markov processes whose parameters are functions. When the functions are MILP-representable, which encompasses many machine learning models, the overall verification problem is equivalent to solving a bilinear program, and our decomposition scheme provides a practical, scalable way of solving such a program.
- Chapter 4 presented the first cost-effectiveness analysis of drone-delivered defibrillators, finding that drones are a cost-effective intervention. We also applied the algorithms developed in Chapter 2 and 3 to further argue that drones are cost-effective over a range of modeling assumptions.

Ultimately, this dissertation shows that the powerful tools developed in automated reasoning, i.e., those that can provide exact proofs or guarantees of the behavior of a system, can be applied to the study of Markov processes. In Chapter 2, we used symbolic methods, while in Chapter 3, we used an optimization-based approach. In a larger sense, it also shows that the exact guarantees provided by automated reasoning are not just of theoretical interest: they have clearly delineable benefits to the very practical problem of health economic analysis. In keeping with this desire to solve real problems, the algorithms developed in this thesis have been packaged into two software libraries, `markovag` and `markovml`, which can be easily used by practitioners. The hope is that these tools will empower researchers, engineers, and policymakers to make robust, transparent, and trustworthy decisions whenever there is uncertainty in stochastic models.

Bibliography

Niels Henrik Abel. *Mémoire sur les équations algébriques, où on demontre l'impossibilité de la résolution de l'équation générale du cinquième dégré.* 1824.

Oguzhan Alagoz, Lisa M Maillart, Andrew J Schaefer, and Mark S Roberts. Determining the acceptance of cadaveric livers using an implicit model of the waiting list. *Operations Research*, 55 (1):24–36, 2007.

Lars W Andersen, Mathias J Holmberg, Asger Granfeldt, Lyndon P James, and Lisa Caulley. Cost-effectiveness of public automated external defibrillators. *Resuscitation*, 138:250–258, 2019.

Ross Anderson, Joey Huchette, Will Ma, Christian Tjandraatmadja, and Juan Pablo Vielma. Strong mixed-integer programming formulations for trained neural networks. *Mathematical Programming*, 183(1):3–39, 2020.

Lazaros Andronis, Pelham Barton, and Stirling Bryan. Sensitivity analysis in economic evaluation: an audit of nice current practice and a review of its use and value in decision-making. *Health technology assessment (Winchester, England)*, 13(29):iii–ix, 2009.

Tu Anh-Nguyen and Joey Huchette. Neural network verification as piecewise linear optimization: Formulations for the composition of staircase functions. *arXiv preprint arXiv:2211.14706*, 2022.

Thom Badings, Sebastian Junges, Ahmadreza Marandi, Ufuk Topcu, and Nils Jansen. Efficient sensitivity analysis for parametric robust markov chains. In *International Conference on Computer Aided Verification*, pages 62–85. Springer, 2023.

Christel Baier and Joost-Pieter Katoen. *Principles of model checking.* MIT press, 2008.

Gianluca Baio and A Philip Dawid. Probabilistic sensitivity analysis in health economics. *Statistical methods in medical research*, 24(6):615–634, 2015.

Bank of Canada. Inflation calculator. `https://www.bankofcanada.ca/rates/related/inflation-calculator/`. Accessed October 9, 2023.

Haniel Barbosa, Clark W. Barrett, Martin Brain, Gereon Kremer, Hanna Lachnitt, Makai Mann, Abdalrhman Mohamed, Mudathir Mohamed, Aina Niemetz, Andres Nötzli, Alex Ozdemir, Mathias Preiner, Andrew Reynolds, Ying Sheng, Cesare Tinelli, and Yoni Zohar. cvc5: A versatile and industrial-strength SMT solver. In Dana Fisman and Grigore Rosu, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 28th International Conference, TACAS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS*

2022, Munich, Germany, April 2-7, 2022, Proceedings, Part I, volume 13243 of *Lecture Notes in Computer Science*, pages 415–442. Springer, 2022. doi: 10.1007/978-3-030-99524-9_24. URL `https://doi.org/10.1007/978-3-030-99524-9_24`.

Richard E Barlow and Frank Proschan. *Mathematical theory of reliability.* SIAM, 1996.

Saugata Basu, Richard Pollack, and Marie-Françoise Roy. *Algorithms in Real Algebraic Geometry.* Springer Berlin, Heidelberg, 2006.

Jan Bauer, Dieter Moormann, Reinhard Strametz, and David A Groneberg. Development of unmanned aerial vehicle (uav) networks delivering early defibrillation for out-of-hospital cardiac arrests (ohca) in areas lacking timely access to emergency medical services (ems) in germany: a comparative economic study. *BMJ Open*, 11(1):e043791, 2021.

Stuart Berger. Cpr and aeds save lives: insuring cpr–aed education and cpr–aed access in schools. *Current opinion in pediatrics*, 32(5):641–645, 2020.

Abraham Berman and Robert J Plemmons. *Nonnegative matrices in the mathematical sciences.* SIAM, 1994.

Hans Blanc and D den Hertog. On markov chains with uncertain data. Technical report, Tilburg University, School of Economics and Management, 2008.

Grigoriy Blekherman, Pablo A Parrilo, and Rekha R Thomas. *Semidefinite optimization and convex algebraic geometry.* SIAM, 2012.

Brittany Bogle, Wayne D Rosamond, Kyle T Snyder, and Jessica K Zègre-Hemsey. The case for drone-assisted emergency response to cardiac arrest: an optimized statewide deployment approach. *North Carolina medical journal*, 80(4):204, 2019.

Justin J Boutilier and Timothy CY Chan. Drone network design for cardiac arrest response. *Manufacturing & Service Operations Management*, 24(5):2407–2424, 2022.

Justin J Boutilier, Steven C Brooks, Alyf Janmohamed, Adam Byers, Jason E Buick, Cathy Zhan, Angela P Schoellig, Sheldon Cheskes, Laurie J Morrison, and Timothy CY Chan. Optimizing a drone network to deliver automated external defibrillators. *Circulation*, 135(25):2454–2465, 2017.

PR Breeze, C Thomas, H Squires, A Brennan, C Greaves, P Diggle, E Brunner, A Tabak, L Preston, and J Chilcott. Cost-effectiveness of population-based, community, workplace and individual policies for diabetes prevention in the uk. *Diabetic Medicine*, 34(8):1136–1144, 2017.

Andrew Briggs, Mark Sculpher, and Martin Buxton. Uncertainty in the economic evaluation of health care technologies: the role of sensitivity analysis. *Health economics*, 3(2):95–104, 1994.

Andrew H Briggs, Milton C Weinstein, Elisabeth AL Fenwick, Jonathan Karnon, Mark J Sculpher, A David Paltiel, ISPOR-SMDM Modeling Good Research Practices Task Force, et al. Model parameter estimation and uncertainty: a report of the ispor-smdm modeling good research practices task force-6. *Value in Health*, 15(6):835–842, 2012.

Christopher W. Brown. *Solution formula construction for truth invariant cad's.* PhD thesis, USA, 1999.

Christopher W Brown. An overview of qepcad b: a tool for real quantifier elimination and formula simplification. *Journal of Japan Society for Symbolic and Algebraic Computation*, 10(1):13–22, 2003.

John Canny. A new algebraic method for robot motion planning and real geometry. In *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, pages 39–48. IEEE, 1987.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. Ieee, 2017.

Andrea Carta and Claudio Conversano. On the use of markov models in pharmacoeconomics: pros and cons and implications for policy makers. *Frontiers in public health*, 8:569500, 2020.

Hal Caswell. Sensitivity analysis of discrete markov chains via matrix calculus. *Linear Algebra and its Applications*, 438(4):1727–1745, 2013.

Hal Caswell. Sensitivity analysis of discrete markov chains. *Sensitivity Analysis: Matrix Methods in Demography and Ecology*, pages 255–280, 2019.

Timothy CY Chan and Muhammad Maaz. Exact sensitivity analysis of markov reward processes via algebraic geometry. *arXiv preprint arXiv:2410.05471*, 2024.

Taolue Chen, Ernst Moritz Hahn, Tingting Han, Marta Kwiatkowska, Hongyang Qu, and Lijun Zhang. Model repair for markov decision processes. In *2013 International Symposium on Theoretical Aspects of Software Engineering*, pages 85–92. IEEE, 2013.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

Chih-Hong Cheng, Georg Nührenberg, and Harald Ruess. Maximum resilience of artificial neural networks. In *Automated Technology for Verification and Analysis: 15th International Symposium, ATVA 2017, Pune, India, October 3–6, 2017, Proceedings 15*, pages 251–268. Springer, 2017.

Sheldon Cheskes, Shelley L McLeod, Michael Nolan, Paul Snobelen, Christian Vaillancourt, Steven C Brooks, Katie N Dainty, Timothy CY Chan, and Ian R Drennan. Improving access to automated external defibrillators in rural and remote settings: a drone delivery feasibility study. *Journal of the American Heart Association*, 9(14):e016687, 2020.

Richard Chocron, Carol Fahrenbruch, Lihua Yin, Sally Guan, Christopher Drucker, Jenny Shin, Mickey Eisenberg, Neal A Chatterjee, Peter J Kudenchuk, and Thomas Rea. Association between functional status at hospital discharge and long-term survival after out-of-hospital-cardiac-arrest. *Resuscitation*, 164:30–37, 2021.

Andreas Claesson, D Fredman, Leif Svensson, Mattias Ringh, Jacob Hollenberg, Per Nordberg, M Rosenqvist, Therese Djarv, S Österberg, Josefin Lennartsson, et al. Unmanned aerial vehicles (drones) in out-of-hospital-cardiac-arrest. *Scandinavian journal of trauma, resuscitation and emergency medicine*, 24:1–9, 2016.

William Cohen, Pradeep Ravikumar, and Stephen Fienberg. A comparison of string metrics for matching names and records. In *Kdd workshop on data cleaning and object consolidation*, volume 3, pages 73–78, 2003.

George E Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition–preliminary report. *ACM SIGSAM Bulletin*, 8(3):80–90, 1974.

George E Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition: a synopsis. *ACM SIGSAM Bulletin*, 10(1):10–12, 1976.

Liyi Dai. Sensitivity analysis of stationary performance measures for markov chains. *Mathematical and computer modelling*, 23(11-12):143–160, 1996.

George B Dantzig, Alex Orden, Philip Wolfe, et al. The generalized simplex method for minimizing a linear form under linear inequality restraints. *Pacific Journal of Mathematics*, 5(2):183–195, 1955.

Gert De Cooman, Filip Hermans, and Erik Quaeghebeur. Sensitivity analysis for finite markov chains in discrete time. *arXiv preprint arXiv:1408.2029*, 2014.

Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer, 2008.

Leonardo De Moura and Nikolaj Bjørner. Satisfiability modulo theories: introduction and applications. *Communications of the ACM*, 54(9):69–77, 2011.

Leonardo De Moura and Sebastian Ullrich. The lean 4 theorem prover and programming language. In *Automated Deduction–CADE 28: 28th International Conference on Automated Deduction, Virtual Event, July 12–15, 2021, Proceedings 28*, pages 625–635. Springer, 2021.

Erick Delage and Shie Mannor. Percentile optimization for markov decision processes with parameter uncertainty. *Operations research*, 58(1):203–213, 2010.

Benoît Delahaye, Didier Lime, and Laure Petrucci. Parameter synthesis for parametric interval markov chains. In *International Conference on Verification, Model Checking, and Abstract Interpretation*, pages 372–390. Springer, 2015.

Sarah Dewilde, Lieven Annemans, Andre Peeters, Dimitri Hemelsoet, Yves Vandermeeren, Philippe Desfontaines, Raf Brouns, Geert Vanhooren, Patrick Cras, Boudewijn Michielsens, et al. Modified rankin scale as a determinant of direct medical costs after stroke. *International Journal of Stroke*, 12(4):392–400, 2017.

Matthew England, Russell Bradford, and James H Davenport. Improving the use of equational constraints in cylindrical algebraic decomposition. In *Proceedings of the 2015 ACM on International Symposium on Symbolic and Algebraic Computation*, pages 165–172, 2015.

Matthew England, Russell Bradford, and James H Davenport. Cylindrical algebraic decomposition with equational constraints. *Journal of Symbolic Computation*, 100:38–71, 2020.

Stephanie Fledsberg and Emelie Pauli. Drönarlevererade hjärtstartare i vgr [drone delivered defibrillator in vgr]. Sweden. Västra Götalandsregionen [West Gotaland Region], Aug 2023.

Joseph Fourier. Histoire de l'académie, partie mathématique (1824). *Mémoires de l'Académie des sciences de l'Institut de France*, 7, 1827.

Guillaume Geri, Carol Fahrenbruch, Hendrika Meischke, Ian Painter, Lindsay White, Thomas D Rea, and Marcia R Weaver. Effects of bystander cpr following out-of-hospital cardiac arrest on hospital costs and long-term survival. *Resuscitation*, 115:129–134, 2017.

Guillaume Geri, Damon C Scales, Maria Koh, Harindra C Wijeysundera, Steve Lin, Michael Feldman, Sheldon Cheskes, Paul Dorian, Wanrudee Isaranuwatchai, Laurie J Morrison, et al. Healthcare costs and resource utilization associated with treatment of out-of-hospital cardiac arrest. *Resuscitation*, 153:234–242, 2020.

Joel Goh, Mohsen Bayati, Stefanos A Zenios, Sundeep Singh, and David Moore. Data uncertainty in markov chains: Application to cost-effectiveness analyses of medical innovations. *Operations Research*, 66(3):697–715, 2018.

Marthe R Gold. *Cost-effectiveness in health and medicine.* Oxford university press, 1996.

Vineet Goyal and Julien Grand-Clément. Robust markov decision processes: Beyond rectangularity. *Mathematics of Operations Research*, 48(1):203–226, 2023.

Julien Grand-Clément and Marek Petrik. On the convex formulations of robust markov decision processes. *Mathematics of Operations Research*, 2024.

Joseph F Grcar. Mathematicians of gaussian elimination. *Notices of the AMS*, 58(6):782–792, 2011.

LLC Gurobi Optimization. Gurobi machine learning, 2024. URL `https://pypi.org/project/gurobi-machinelearning/`. Python package for integrating regression models into optimization problems.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2024. URL `https://www.gurobi.com`.

David J Hand and Robert J Till. A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine learning*, 45:171–186, 2001.

Carolina Malta Hansen, Kristian Kragholm, David A Pearson, Clark Tyson, Lisa Monk, Brent Myers, Darrell Nelson, Matthew E Dupre, Emil L Fosbøl, James G Jollis, et al. Association of bystander and first-responder intervention with survival after out-of-hospital cardiac arrest in north carolina, 2010-2013. *Jama*, 314(3):255–264, 2015.

Hans Hansson and Bengt Jonsson. A logic for reasoning about time and reliability. *Formal aspects of computing*, 6:512–535, 1994.

Heart and Stroke Foundation of Canada. Saving lives. `https://www.heartandstroke.ca/what-we-do/our-impact/saving-lives`, 2023. Accessed July 20, 2023.

Jerris R Hedges, Ruchir Sehra, Jonathan W Van Zile, Andrew R Anton, Lois A Bosken, Robert E O'Connor, Richard Moore, Judy L Powell, and Mary Ann McBurnie. Automated external defibrillator program does not impair cardiopulmonary resuscitation initiation in the public access defibrillation trial. *Academic emergency medicine*, 13(6):659–665, 2006.

Patrick Henriksen and Alessio Lomuscio. Efficient neural network verification via adaptive refinement and adversarial search. In *ECAI 2020*, pages 2513–2520. IOS Press, 2020.

Patrick Henriksen and Alessio Lomuscio. Deepsplit: An efficient splitting method for neural network verification via indirect effect analysis. In *IJCAI*, pages 2549–2555, 2021.

Holger Hermanns, Joost-Pieter Katoen, Joachim Meyer-Kayser, and Markus Siegle. A markov chain model checker. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 347–362. Springer, 2000.

Filip Hermans and Gert De Cooman. Characterisation of ergodic upper transition operators. *International Journal of Approximate Reasoning*, 53(4):573–583, 2012.

Hoon Hong. An improvement of the projection operator in cylindrical algebraic decomposition. In *Proceedings of the international symposium on Symbolic and algebraic computation*, pages 261–264, 1990.

Jaroslav Horáček. *Interval linear and nonlinear systems.* Phd thesis, Univerzita Karlova, Matematicko-fyzikální fakulta, Prague, Czech Republic, 2019.

Reiner Horst and Panos M Pardalos. *Handbook of global optimization*, volume 2. Springer Science & Business Media, 2013.

Philip Hougaard. Multi-state models: a review. *Lifetime data analysis*, 5:239–264, 1999.

Garud N Iyengar. Robust dynamic programming. *Mathematics of Operations Research*, 30(2): 257–280, 2005.

Rahul Jain, Michael Grabner, and Eberechukwu Onukwugha. Sensitivity analysis in cost-effectiveness studies: from guidelines to practice. *Pharmacoeconomics*, 29:297–314, 2011.

R. G. Jeroslow and J. K. Lowe. *Modelling with integer variables*, pages 167–184. Springer Berlin Heidelberg, Berlin, Heidelberg, 1984.

Dejan Jovanović and Leonardo De Moura. Solving non-linear arithmetic. *ACM Communications in Computer Algebra*, 46(3/4):104–105, 2013.

Sebastian Junges. *Parameter synthesis in Markov models.* PhD thesis, Dissertation, RWTH Aachen University, 2020, 2020.

Sebastian Junges, Erika Ábrahám, Christian Hensel, Nils Jansen, Joost-Pieter Katoen, Tim Quatmann, and Matthias Volk. Parameter synthesis for markov models: covering the parameter space. *Formal Methods in System Design*, 62(1):181–259, 2024.

Lena Karlsson, Carolina Malta Hansen, Mads Wissenberg, Steen Møller Hansen, Freddy K Lippert, Shahzleen Rajan, Kristian Kragholm, Sidsel G Møller, Kathrine Bach Søndergaard, Gunnar H Gislason, et al. Automated external defibrillator accessibility is crucial for bystander defibrillation and survival: a registry-based study. *Resuscitation*, 136:30–37, 2019.

J-P Katoen, Maneesh Khattri, and IS Zapreevt. A markov reward model checker. In *Second International Conference on the Quantitative Evaluation of Systems (QEST'05)*, pages 243–244. IEEE, 2005.

Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I 30*, pages 97–117. Springer, 2017.

Guy Katz, Derek A Huang, Duligur Ibeling, Kyle Julian, Christopher Lazarus, Rachel Lim, Parth Shah, Shantanu Thakoor, Haoze Wu, Aleksandar Zeljić, et al. The marabou framework for verification and analysis of deep neural networks. In *Computer Aided Verification: 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part I 31*, pages 443–452. Springer, 2019.

Suhas Kotha, Christopher Brix, J. Zico Kolter, Krishnamurthy Dvijotham, and Huan Zhang. Provably bounding neural network preimages. In A. Oh, T. Neumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 80270–80290, 2023.

Gereon Kremer and Erika Ábrahám. Fully incremental cylindrical algebraic decomposition. *Journal of Symbolic Computation*, 100:11–37, 2020.

Gereon Kremer, Andrew Reynolds, Clark Barrett, and Cesare Tinelli. Cooperating techniques for solving nonlinear real arithmetic in the cvc5 smt solver (system description). In *International Joint Conference on Automated Reasoning*, pages 95–105. Springer International Publishing Cham, 2022.

Eline M Krijkamp, Fernando Alarid-Escudero, Eva A Enns, Hawre J Jalal, MG Myriam Hunink, and Petros Pechlivanoglou. Microsimulation modeling for health decision sciences using r: a tutorial. *Medical Decision Making*, 38(3):400–422, 2018.

Jan Kronqvist, Ruth Misener, and Calvin Tsay. Between steps: Intermediate relaxations between big-m and convex hull formulations. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 299–314. Springer, 2021.

Marta Kwiatkowska, Gethin Norman, and David Parker. Prism: Probabilistic symbolic model checker. In *International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, pages 200–204. Springer, 2002.

Richard Lassaigne and Sylvain Peyronnet. Approximate verification of probabilistic systems. In *Joint International Workshop von Process Algebra and Probabilistic Methods, Performance Modeling and Verification*, pages 213–214. Springer, 2002.

Jean B Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on optimization*, 11(3):796–817, 2001.

Yujin Lee, Dariush Mozaffarian, Stephen Sy, Junxiu Liu, Parke E Wilde, Matti Marklund, Shafika Abrahams-Gessel, Thomas A Gaziano, and Renata Micha. Health impact and cost-effectiveness of volume, tiered, and absolute sugar content sugar-sweetened beverage tax policies in the united states: a microsimulation study. *Circulation*, 142(6):523–534, 2020.

Steve Lin, Laurie J Morrison, and Steven C Brooks. Development of a data dictionary for the strategies for post arrest resuscitation care (sparc) network for post cardiac arrest research. *Resuscitation*, 82(4):419–422, 2011.

Stanislaw Lojasiewicz. Ensembles semi-analytiques. *Lectures Notes IHES (Bures-sur-Yvette)*, 1965.

Yin Lou, Rich Caruana, Johannes Gehrke, and Giles Hooker. Accurate intelligible models with pairwise interactions. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 623–631, 2013.

Muhammad Maaz and Timothy CY Chan. Formal verification of markov processes with learned parameters. *arXiv preprint arXiv:2501.15767*, 2025.

Muhammad Maaz, K. H. Benjamin Leung, Justin J. Boutilier, Sze chuan Suen, Paul Dorian, Laurie J. Morrison, Damon C. Scales, Sheldon Cheskes, and Timothy C. Y. Chan. Cost-effectiveness of drone-delivered automated external defibrillators for cardiac arrest. *Under review*, 2024.

Muhammad Maaz, KH Benjamin Leung, Justin J Boutilier, Sze-chuan Suen, Paul Dorian, Laurie J Morrison, Damon C Scales, Sheldon Cheskes, and Timothy CY Chan. Cost-effectiveness of drone-delivered automated external defibrillators for cardiac arrest. *Resuscitation*, 209:110552, 2025.

Peter Marbach and John N Tsitsiklis. Simulation-based optimization of markov reward processes. *IEEE Transactions on Automatic Control*, 46(2):191–209, 2001.

Peter Marbach and John N Tsitsiklis. Approximate gradient methods in policy-space optimization of markov reward processes. *Discrete Event Dynamic Systems*, 13:111–148, 2003.

Garth P McCormick. Computability of global solutions to factorable nonconvex programs: Part i—convex underestimating problems. *Mathematical programming*, 10(1):147–175, 1976.

Elly Mertens, Els Genbrugge, Junior Ocira, and José L Peñalvo. Microsimulation modeling in food policy: A scoping review of methodological aspects. *Advances in Nutrition*, 13(2):621–632, 2022.

Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, January 2017. ISSN 2376-5992. doi: 10.7717/peerj-cs.103. URL `https://doi.org/10.7717/peerj-cs.103`.

J Donald Monk. *Introduction to set theory.* McGraw-Hill, 1969.

Arnold Neumaier. New techniques for the analysis of linear interval equations. *Linear Algebra and its Applications*, 58:273–325, 1984.

Arnab Nilim and Laurent El Ghaoui. Robust solutions to markov decision problems with uncertain transition matrices. *Operations Research*, 53(5):780–798, 2005.

Michael P Owen, Adam Panken, Robert Moss, Luis Alvarez, and Charles Leeper. ACAS Xu: Integrated collision avoidance and detect and avoid capability for uas. In *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*, pages 1–10. IEEE, 2019.

Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity.* Courier Corporation, 2013.

Puja B Parikh, Lukman Hassan, Asem Qadeer, and Jignesh K Patel. Association between sex and mortality in adults with in-hospital and out-of-hospital cardiac arrest: A systematic review and meta-analysis. *Resuscitation*, 155:119–124, 2020.

Pablo A Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Mathematical programming*, 96:293–320, 2003.

Pablo A Parrilo and Rekha R Thomas. Sum of squares: theory and applications. *AMS short course, Baltimore, MD, USA*, 77, 2019.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Laure Petrucci and Jaco van De Pol. Parameter synthesis algorithms for parametric interval markov chains. In *International Conference on Formal Techniques for Distributed Objects, Components, and Systems*, pages 121–140. Springer, 2018.

Frederic Portoraro. Automated Reasoning. In Edward N. Zalta and Uri Nodelman, editors, *The Stanford Encyclopedia of Philosophy.* Metaphysics Research Lab, Stanford University, Summer 2025 edition, 2025.

Aaron Pulver and Ran Wei. Optimizing the spatial location of medical drones. *Applied geography*, 90:9–16, 2018.

Aaron Pulver, Ran Wei, and Clay Mann. Locating aed enabled medical drones to enhance cardiac arrest response times. *Prehospital Emergency Care*, 20(3):378–389, 2016.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming.* John Wiley & Sons, 2014.

Srikant Rangaraju, Diogo Haussen, Raul G Nogueira, Fadi Nahab, and Michael Frankel. Comparison of 3-month stroke disability and quality of life across modified rankin scale categories. *Interventional neurology*, 6(1-2):36–41, 2017.

Marvin Rausand and Arnljot Hoyland. *System reliability theory: models, statistical methods, and applications*, volume 396. John Wiley & Sons, 2003.

Johann WA Röper, Katharina Fischer, Mina Carolina Baumgarten, Karl Christian Thies, Klaus Hahnenkamp, and Steffen Fleßa. Can drones save lives and money? an economic evaluation of airborne delivery of automated external defibrillators. *The European Journal of Health Economics*, 24(7):1141–1150, 2023.

Jeffrey S Rosenthal. *A first look at stochastic processes*. World Scientific, 2019.

Luke Rudmik and Michael Drummond. Health economic evaluation: important principles and methodology. *The Laryngoscope*, 123(6):1341–1347, 2013.

Paolo Ruffini. *La teoria generale delle equazioni in cui é provato che la soluzione algebrica di equazioni di grado maggiore di 4 é impossibile*. 1799.

Comilla Sasson, Mary AM Rogers, Jason Dahl, and Arthur L Kellermann. Predictors of survival from out-of-hospital cardiac arrest: a systematic review and meta-analysis. *Circulation: Cardiovascular Quality and Outcomes*, 3(1):63–81, 2010.

Sofia Schierbeck, Jacob Hollenberg, Anette Nord, Leif Svensson, Per Nordberg, Mattias Ringh, Sune Forsberg, Peter Lundgren, Christer Axelsson, and Andreas Claesson. Automated external defibrillators delivered by drones to patients with suspected out-of-hospital cardiac arrest. *European Heart Journal*, 43(15):1478–1487, 2022a.

Sofia Schierbeck, Leif Svensson, and Andreas Claesson. Use of a drone-delivered automated external defibrillator in an out-of-hospital cardiac arrest. *New England Journal of Medicine*, 386(20): 1953–1954, 2022b.

Sofia Schierbeck, Anette Nord, Leif Svensson, Mattias Ringh, Per Nordberg, Jacob Hollenberg, Peter Lundgren, Fredrik Folke, Martin Jonsson, Sune Forsberg, et al. Drone delivery of automated external defibrillators compared with ambulance arrival in real-life suspected out-of-hospital cardiac arrests: a prospective observational study in sweden. *The Lancet Digital Health*, 5(12):e862–e871, 2023.

Jacob T Schwartz and Micha Sharir. Algorithmic motion planning in robotics. In *Algorithms and Complexity*, pages 391–430. Elsevier, 1990.

Koushik Sen, Mahesh Viswanathan, and Gul Agha. On statistical model checking of stochastic systems. In *Computer Aided Verification: 17th International Conference, CAV 2005, Edinburgh, Scotland, UK, July 6-10, 2005. Proceedings 17*, pages 266–280. Springer, 2005.

Koushik Sen, Mahesh Viswanathan, and Gul Agha. Model-checking markov chains in the presence of uncertainties. In *Tools and Algorithms for the Construction and Analysis of Systems: 12th International Conference, TACAS 2006, Held as Part of the Joint European Conferences on Theory* *and Practice of Software, ETAPS 2006, Vienna, Austria, March 25-April 2, 2006. Proceedings 12*, pages 394–410. Springer, 2006.

Frank A Sonnenberg and J Robert Beck. Markov models in medical decision making: a practical guide. *Medical decision making*, 13(4):322–338, 1993.

Monique A Starks, Audrey L Blewer, Edward Sharpe, Lee Van Vleet, Jennie Riley, Evan Arnold, Joseph Slattery, Anjni Joiner, Daniel M Buckland, Jinny Ye, et al. Bystander performance during simulated drone delivery of an aed for mock out-of-hospital cardiac arrest. *Journal of the American College of Cardiology*, 75(11_Supplement_1):303–303, 2020.

Statistics Canada. 2021 census - attribute domain values. `https://www12.statcan.gc.ca/census-recensement/2021/geo/ref/domain-domaine/index2021-eng.cfm?lang=e&id=POPCTRRAclass`, 2022. Accessed October 9, 2023.

Statistics Canada. Dictionary, census of population, 2016 - population centre (popctr). `https://www12.statcan.gc.ca/census-recensement/2016/ref/dict/geo049a-eng.cfm`, 2023a. Accessed October 9, 2023.

Statistics Canada. Population centre boundary file. `https://www150.statcan.gc.ca/n1/en/catalogue/92-166-X`, 2023b. Accessed October 9, 2023.

Statistics Canada. Census tract boundary files. `https://www150.statcan.gc.ca/n1/en/catalogue/92-168-X`, 2023c. Accessed September 4, 2023.

William J Stewart. *Introduction to the numerical solution of Markov chains*. Princeton University Press, 2021.

Gilbert Strang. *Introduction to linear algebra*. SIAM, 2022.

Adam Strzeboński. Computation with semialgebraic sets represented by cylindrical algebraic formulas. In *Proceedings of the 2010 International Symposium on Symbolic and Algebraic Computation*, pages 61–68, 2010.

Christopher LF Sun, Lena Karlsson, Christian Torp-Pedersen, Laurie J Morrison, Fredrik Folke, and Timothy CY Chan. Spatiotemporal aed optimization is generalizable. *Resuscitation*, 131: 101–107, 2018.

Alfred Tarski. A decision method for elementary algebra and geometry. In *Quantifier elimination and cylindrical algebraic decomposition*, pages 24–84. Springer, 1951.

Christian Tjandraatmadja, Ross Anderson, Joey Huchette, Will Ma, Krunal Kishor Patel, and Juan Pablo Vielma. The convex relaxation barrier, revisited: Tightened single-neuron relaxations for neural network verification. *Advances in Neural Information Processing Systems*, 33:21675–21686, 2020.

Vincent Tjeng, Kai Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. *arXiv preprint arXiv:1711.07356*, 2017.

TreeAge Software. Treeage pro 2021, r1. `http://www.treeage.com`, 2021. TreeAge Software, Williamstown, MA.

Unity Health Toronto. Rescu epistry. `https://research.unityhealth.to/research-programs/rescu/our-research/epistry/`, 2023. Accessed September 23, 2023.

Rick A Vreman, Joost W Geenen, Saskia Knies, Aukje K Mantel-Teeuwisse, Hubert GM Leufkens, and Wim G Goettsch. The application and implications of novel deterministic sensitivity analysis methods. *Pharmacoeconomics*, 39:1–17, 2021.

Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and J Zico Kolter. Beta-CROWN: Efficient bound propagation with per-neuron split constraints for complete and incomplete neural network verification. *Advances in Neural Information Processing Systems*, 34, 2021.

Wolfram Wiesemann, Daniel Kuhn, and Berç Rustem. Robust markov decision processes. *Mathematics of Operations Research*, 38(1):153–183, 2013.

Parke Wilde, Yue Huang, Stephen Sy, Shafika Abrahams-Gessel, Thiago Veiga Jardim, Robert Paarlberg, Dariush Mozaffarian, Renata Micha, and Thomas Gaziano. Cost-effectiveness of a us national sugar-sweetened beverage tax with a multistakeholder approach: who pays and who benefits. *American journal of public health*, 109(2):276–284, 2019.

JT Lindsay Wilson, Asha Hareendran, Marie Grant, Tracey Baird, Ursula GR Schulz, Keith W Muir, and Ian Bone. Improving the assessment of outcomes in stroke: use of a structured interview to assign grades on the modified rankin scale. *Stroke*, 33(9):2243–2246, 2002.

Wolfram Research. CylindricalDecomposition. `https://reference.wolfram.com/language/ref/CylindricalDecomposition.html`, 2020. Accessed: 03-September-2024.

Haoze Wu, Alex Ozdemir, Aleksandar Zeljic, Kyle Julian, Ahmed Irfan, Divya Gopinath, Sadjad Fouladi, Guy Katz, Corina Pasareanu, and Clark Barrett. Parallelization techniques for verifying neural networks. In *# PLACEHOLDER_PARENT_METADATA_VALUE#*, volume 1, pages 128–137. TU Wien Academic Press, 2020.

Håkan LS Younes. Ymer: A statistical model checker. In *International Conference on Computer Aided Verification*, pages 429–433. Springer, 2005.

Jessica K Zègre-Hemsey, Brittany Bogle, Christopher J Cunningham, Kyle Snyder, and Wayne Rosamond. Delivery of automated external defibrillators (aed) by drones: implications for emergency cardiac care. *Current cardiovascular risk reports*, 12:1–5, 2018.

Jessica K Zègre-Hemsey, Mary E Grewe, Anna M Johnson, Evan Arnold, Christopher J Cunningham, Brittany M Bogle, and Wayne D Rosamond. Delivery of automated external defibrillators via drones in simulated cardiac arrest: users' experiences and the human-drone interaction. *Resuscitation*, 157:83–88, 2020.

Huan Zhang, Shiqi Wang, Kaidi Xu, Linyi Li, Bo Li, Suman Jana, Cho-Jui Hsieh, and J Zico Kolter. General cutting planes for bound-propagation-based neural network verification. *Advances in Neural Information Processing Systems*, 2022a.

Huan Zhang, Shiqi Wang, Kaidi Xu, Yihan Wang, Suman Jana, Cho-Jui Hsieh, and Zico Kolter. A branch and bound framework for stronger adversarial attacks of ReLU networks. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pages 26591–26604, 2022b.

Appendix A

Supplementary Material from Chapter 4

A.1 Supplementary Tables

Table A.1: Summary statistics for final cohort of included records

Variable	**Summary**
Total, count	22017
Age, mean (sd)	68.8 (16.9)
Male, count (%)	14017 (63.7)
Urban, count (%)	20653 (93.8)
Shockable rhythm, count (%)	4599 (20.9)
Witnessed, count (%)	8450 (38.4)
Time to ambulance, mean (sd)	381 (152)
Outcome, count (%)	
Died before hospital	10326 (46.9)
Survived to hospital admission	11691 (53.1)
Died in hospital (mRS 6)	9836 (44.7)
Survived to hospital discharge	1855 (8.4)
mRS 0	536 (2.4)
mRS 1	146 (0.7)
mRS 2	75 (0.3)
mRS 3	48 (0.2)
mRS 4	19 (0.1)
mRS 5	25 (0.1)
Missing mRS	1006 (4.6)

Table A.2: Networks with the highest NMB, at a $150,000 threshold, for each number of drones

Number of drones	**Type of drone network**	**Net monetary benefit, at $150,000 threshold, per-person**
5	Coverage, 6m	156201.62
10	Coverage, 6m	163469.71
15	Mean	169313.60
20	Mean	173138.11
25	Coverage, 4m	175265.39
30	Mean	178679.21
35	Mean	180260.67
40	Mean	182871.48
45	Mean	184180.39
50	Mean	185394.47
55	Mean	187001.12
60	Mean	188183.64
65	Mean	188668.47
70	Mean	189160.87
75	Mean	189721.68
80	Coverage, 3m	190348.81
85	Coverage, 3m	190967.96
90	Coverage, 3m	191201.01
95	Coverage, 3m	191588.56
100	Coverage, 3m	191866.86
105	Mean	192225.93
110	Mean	192705.23
115	Mean	192885.46
120	Coverage, 3m	193134.92
125	Coverage, 3m	193469.80
130	Coverage, 3m	193846.88
135	Coverage, 3m	194008.92
140	Coverage, 3m	194200.89
145	Coverage, 3m	194571.59
150	Mean	194624.41
155	Mean	194821.45
160	Coverage, 3m	195081.29
165	Mean	195283.01
170	Mean	195395.27
175	Mean	195524.56
180	Mean	195657.02
185	Mean	195794.90
190	Mean	195884.43
195	Mean	195929.08
200	Coverage, 3m	196135.43
205	Mean	196169.22
210	Coverage, 3m	196306.54
215	Mean	196385.34

220	Mean	196443.86
225	Mean	196589.95
230	Coverage, 3m	196968.31
235	Coverage, 3m	197375.83
240	Coverage, 3m	197380.19
245	Coverage, 3m	197394.44
250	Coverage, 3m	197387.61
255	Coverage, 3m	197388.66
260	Coverage, 3m	197393.21
265	Coverage, 3m	197399.20
270	Coverage, 3m	197436.18
275	Coverage, 3m	197498.43
280	Coverage, 3m	197505.88
285	Coverage, 2m	197551.72
290	Coverage, 3m	197649.78
295	Coverage, 2m	197692.02
300	Coverage, 2m	197731.54
305	Coverage, 3m	197710.96
310	Coverage, 3m	197846.51
315	Coverage, 2m	197902.04
320	Coverage, 3m	197823.85
325	Coverage, 3m	197792.26
330	Coverage, 3m	197757.80
335	Coverage, 3m	197751.40
340	90th %ile	198353.04
345	90th %ile	198408.97
350	90th %ile	198291.38
355	90th %ile	198401.69
360	90th %ile	198216.41
365	90th %ile	198525.88
370	90th %ile	198199.19
375	90th %ile	198519.74
380	90th %ile	198605.63
385	90th %ile	198393.90
390	90th %ile	198494.31
395	90th %ile	198600.24
400	90th %ile	198203.76
405	90th %ile	198492.44
410	90th %ile	198422.19
415	90th %ile	198502.41
420	90th %ile	198353.04
425	90th %ile	198589.43
430	90th %ile	198523.58
435	90th %ile	198452.64
440	90th %ile	198490.75
445	Mean	198480.16
450	Mean	198451.25

455	90th %ile	198512.38
460	90th %ile	198483.71
465	Mean	198488.48
470	Mean	198489.02
475	Coverage, 7m	198275.42
480	Coverage, 6m	198357.43
485	Mean, 90th %ile (equivalent)	198245.63
490	Mean, 90th %ile (equivalent)	198284.65
495	Mean, 90th %ile (equivalent)	198275.67
500	Coverage, 2m	198268.19
505	Coverage, 2m	198351.81
510	Coverage, 2m	198324.17
515	Coverage, 2m	198290.76
520	Coverage, 2m	198287.09
525	Coverage, 2m	198254.23
530	Coverage, 3m	198227.93
535	Coverage, 1m	198213.30

Table A.3: Patient outcomes, expressed as rates (%), for shockable and/or witnessed populations, for each of the networks on the efficient frontier. Patients are grouped by historical status. A: survivors to hospital admission, D: survivors to hospital discharge, F: neurologically favorable status

Drone network	**Whole population (n=22017)**			**Witnessed (n=8450)**			**Shockable (n=4599)**			**Shockable and witnessed (n=2891)**			**Not shockable nor witnessed (n=11859)**		
	A	*D*	*F*	*A*	*D*	*F*	*A*	*D*	*F*	*A*	*D*	*F*	*A*	*D*	*F*
Standard care	53.10	8.43	7.94	64.21	12.79	12.01	83.06	27.72	26.33	84.85	28.78	27.22	41.30	2.79	2.61
Coverage, 5m, 20 drones	58.63	10.28	9.53	70.09	15.97	14.82	86.30	31.51	29.86	87.95	33.25	31.44	46.89	3.59	3.21
Coverage, 4m, 50 drones	60.76	11.09	10.21	72.06	17.33	16.00	87.45	33.20	31.40	89.03	35.21	33.23	49.26	3.96	3.48
Coverage, 3m, 85 drones	61.75	11.55	10.57	72.97	18.11	16.64	88.04	34.13	32.18	89.61	36.33	34.18	50.34	4.16	3.62
Coverage, 3m, 95 drones	61.83	11.59	10.61	73.06	18.18	16.71	88.07	34.23	32.28	89.66	36.45	34.30	50.43	4.18	3.64
Coverage, 3m, 110 drones	61.96	11.65	10.66	73.19	18.28	16.79	88.11	34.32	32.35	89.72	36.54	34.37	50.58	4.21	3.66
Coverage, 3m, 135 drones	62.25	11.78	10.75	73.47	18.48	16.94	88.31	34.60	32.57	89.92	36.84	34.62	50.89	4.27	3.70
Coverage, 3m, 145 drones	62.33	11.83	10.78	73.55	18.56	17.00	88.37	34.68	32.63	89.99	36.93	34.69	50.98	4.29	3.71
Coverage, 3m, 160 drones	62.40	11.87	10.82	73.62	18.62	17.06	88.41	34.79	32.73	90.02	37.08	34.81	51.06	4.31	3.72
Coverage, 3m, 185 drones	62.53	11.93	10.86	73.76	18.73	17.15	88.52	34.93	32.84	90.12	37.24	34.94	51.18	4.33	3.74
Coverage, 3m, 235 drones	62.86	12.07	10.97	73.93	18.91	17.28	88.62	35.20	33.05	90.15	37.47	35.13	51.63	4.42	3.80
90th %ile, 345 drones	63.06	12.20	11.06	74.09	19.10	17.40	88.77	35.49	33.27	90.27	37.76	35.33	51.87	4.48	3.84
90th %ile, 365 drones	63.09	12.22	11.07	74.11	19.13	17.42	88.80	35.52	33.29	90.29	37.80	35.37	51.90	4.50	3.85
90th %ile, 380 drones	63.11	12.23	11.08	74.12	19.13	17.44	88.82	35.54	33.31	90.29	37.80	35.36	51.93	4.50	3.85
90th %ile, 425 drones	63.15	12.25	11.10	74.14	19.17	17.46	88.82	35.59	33.35	90.31	37.86	35.41	51.99	4.51	3.86
Mean, 470 drones	63.18	12.27	11.11	74.18	19.20	17.48	88.86	35.63	33.38	90.36	37.91	35.46	52.02	4.52	3.86
Coverage, 2m, 505 drones	63.19	12.28	11.11	74.18	19.21	17.49	88.88	35.65	33.39	90.37	37.93	35.47	52.03	4.52	3.86
Coverage, 2m, 520 drones	63.20	12.28	11.11	74.18	19.22	17.50	88.88	35.65	33.40	90.37	37.93	35.48	52.04	4.52	3.87
538 drones	63.20	12.28	11.12	74.18	19.22	17.50	88.88	35.66	33.40	90.37	37.94	35.48	52.04	4.53	3.87

Table A.4: Summary of Prediction Models

Prediction	**Model Type**	**Covariates**
Shockable rhythm	Binary logistic	Age, sex, witnessed, time to ambulance
Survival to hospital	Binary logistic	Age, sex, shockable, witnessed, time to ambulance
mRS at discharge	XGBoost32 (multiclass), hyperparameters: max depth = 3, eta = 0.1, gamma = 0.3, rounds = 50, colsample by tree = 0.8, min child weight = 1, subsample = 0.8, threads = 2	Age, sex, shockable, witnessed, time to ambulance

Table A.5: Comparison of important covariates between patients who survived to discharge and had an mRS versus those who survived but had a missing mRS. Continuous covariates were compared with a two-sample t-test, while proportions were compared with a two-proportion z-test.

Covariate	**Missing (n=1006)**	**Not missing (n=849)**	**p-value**
Age, mean (sd)	60.9 (16.3)	59.3 (15.0)	0.03
Male, n (%)	719 (71.5)	655 (77.1)	0.006
Urban, n (%)	936 (93.0)	791 (93.2)	0.99
Shockable, n (%)	627 (62.3)	648 (76.3)	<0.001
Witnessed, n (%)	499 (49.6)	582 (68.6)	<0.001
Time to ambulance, mean (sd)	376 (165)	355 (134)	0.003

Table A.6: Results of imputation of missing mRS values.

mRS	**Counts of predictions for n=1006 with missing mRS**	**Total counts for whole n=22017 after imputation**
0	928	1464
1	63	209
2	0	75
3	15	63
4	0	19
5	0	25

A.2 Supplementary Figures

Figure A.1: Inclusion and exclusion diagram of dataset

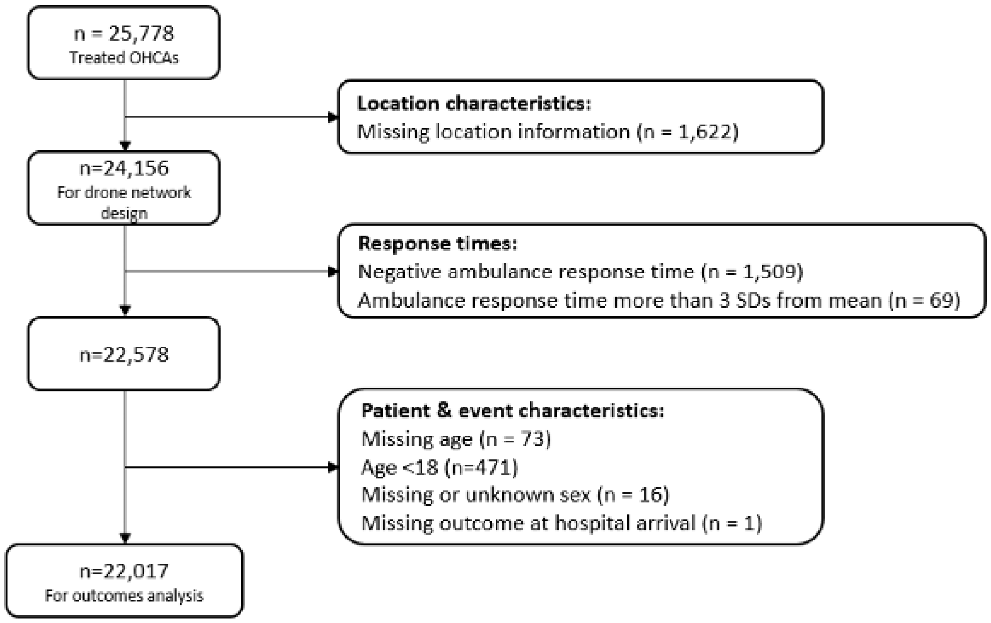

n = 25,778
Treated OHCAs
Location characteristics:
Missing location information (n = 1,622)
n=24,156
For drone network design
Response times:
Negative ambulance response time (n = 1,509)
Ambulance response time more than 3 SDs from mean (n = 69)
n=22,578
Patient & event characteristics:
Missing age (n = 73)
Age <18 (n=471)
Missing or unknown sex (n = 16)
Missing outcome at hospital arrival (n = 1)
n=22,017
For outcomes analysis

Figure A.2: Net monetary benefit of all networks at thresholds of $50k, $100k, and $150k

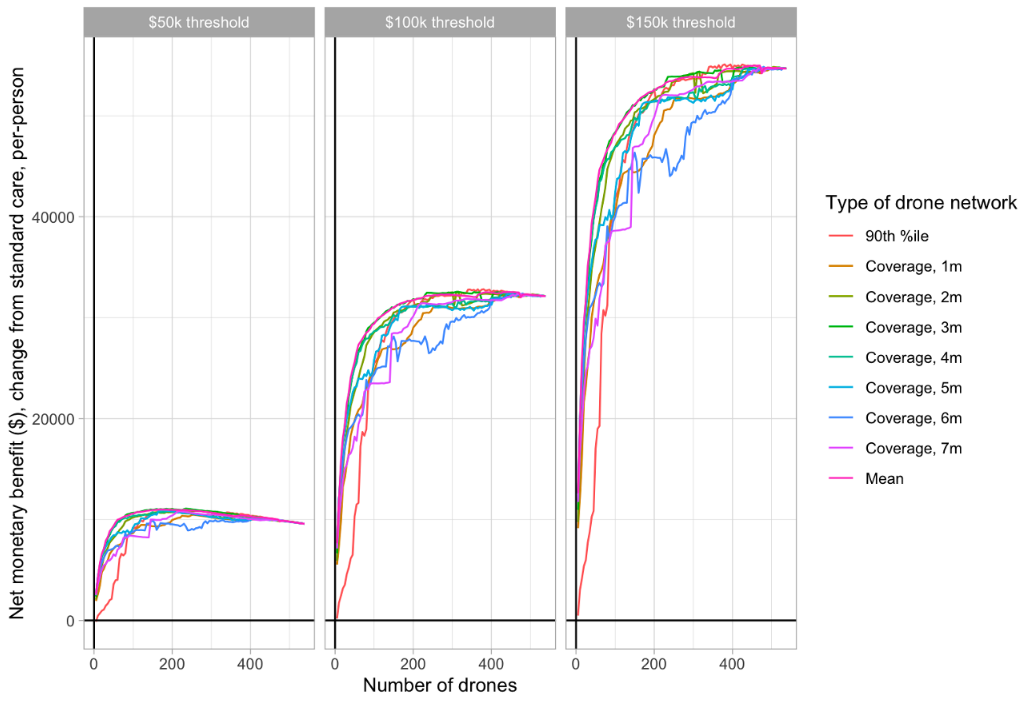


Figure A.3: Subgroup analysis of net monetary benefit at \$150k threshold, for shockable and/or witnessed populations, for drone networks on efficient frontier. Solid lines indicates standard care. Patients are grouped by historical status. Note that per-person normalizes the NMB by dividing by the number of people in that subgroup, in order to make values comparable between subgroups

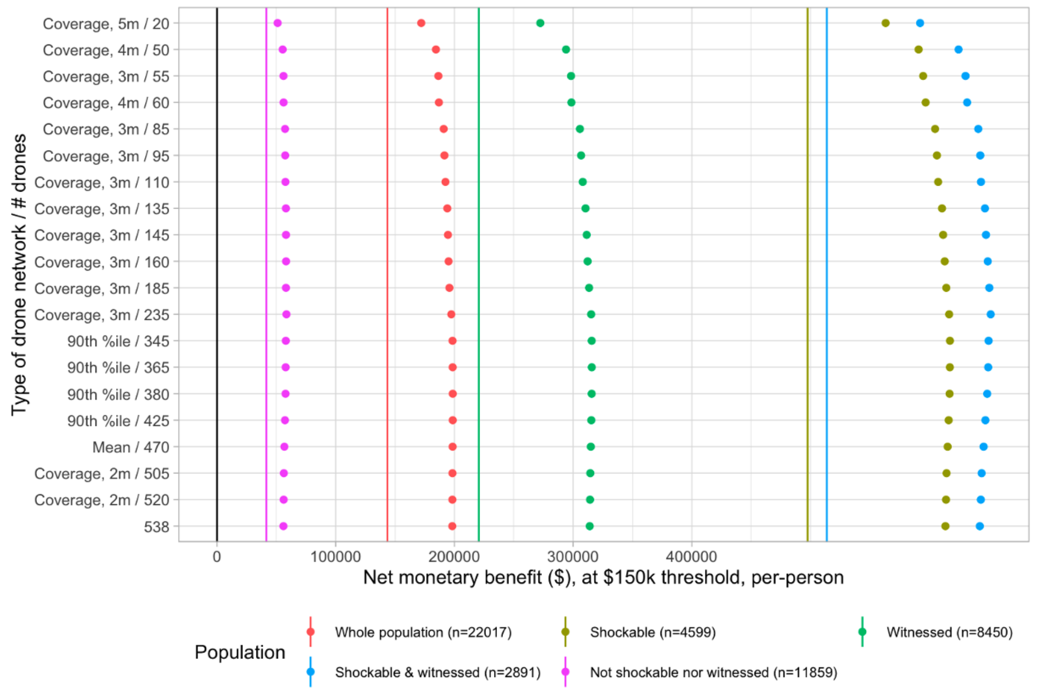


Figure A.4: Tornado plots showing sensitivity of number of survivors to hospital admission to model parameters, for all drone networks on the efficient frontier. Solid line indicates value at the original parameter values. Drone network is labeled as: type of drone network / # drones

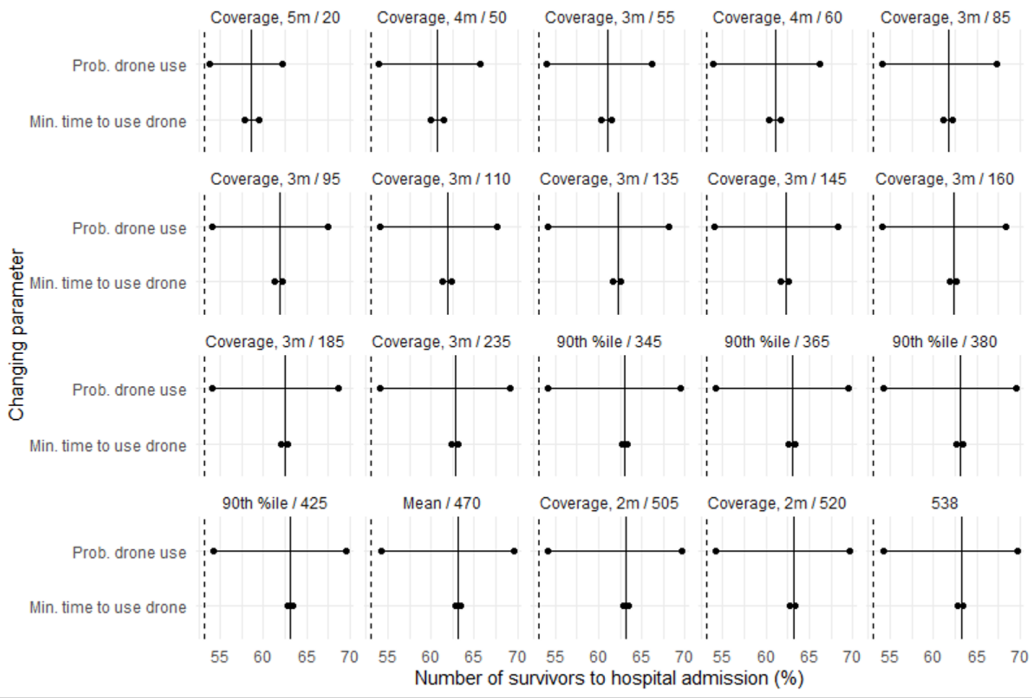


Figure A.5: Tornado plots showing sensitivity of number of survivors to hospital discharge to model parameters, for all drone networks on the efficient frontier. Solid line indicates count at the original parameter values. Drone network is labeled as: type of drone network / # drones

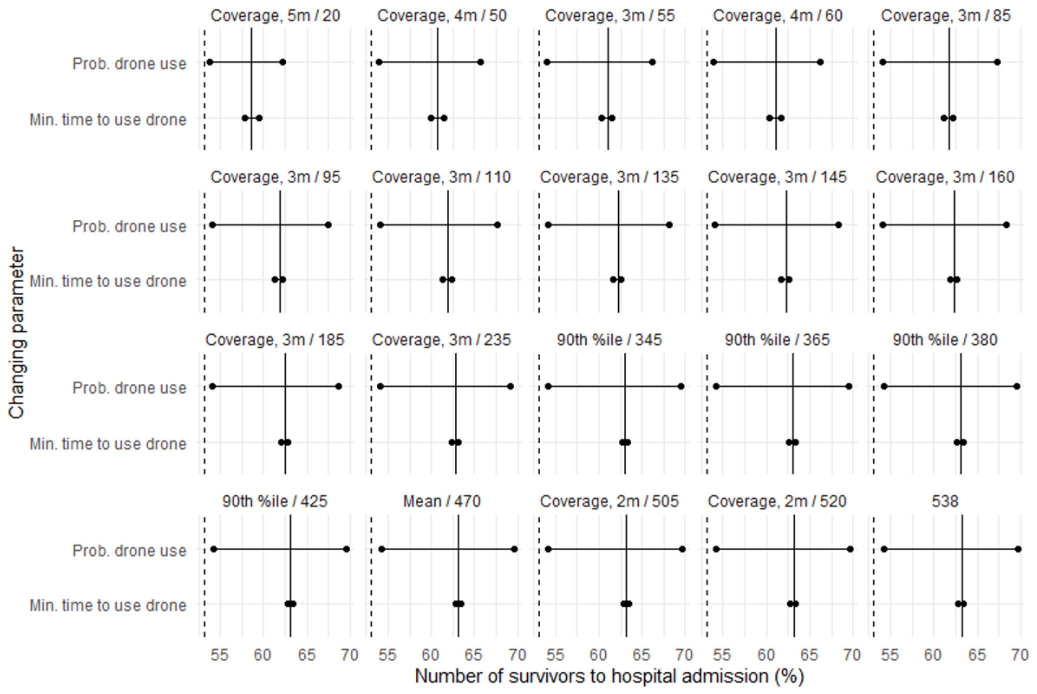


Figure A.6: Tornado plots showing sensitivity of number of survivors with neurologically favorable outcome (mRS 0-2) to model parameters, for all drone networks on the efficient frontier. Solid line indicates count at the original parameter values. Drone network is labeled as: type of drone network / # drones

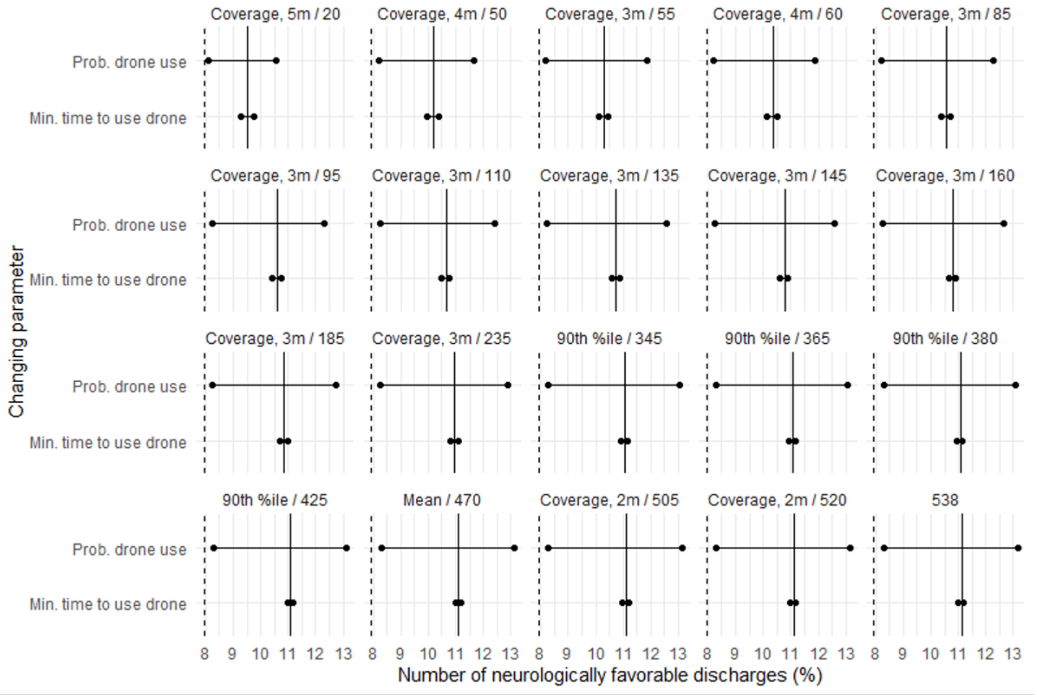
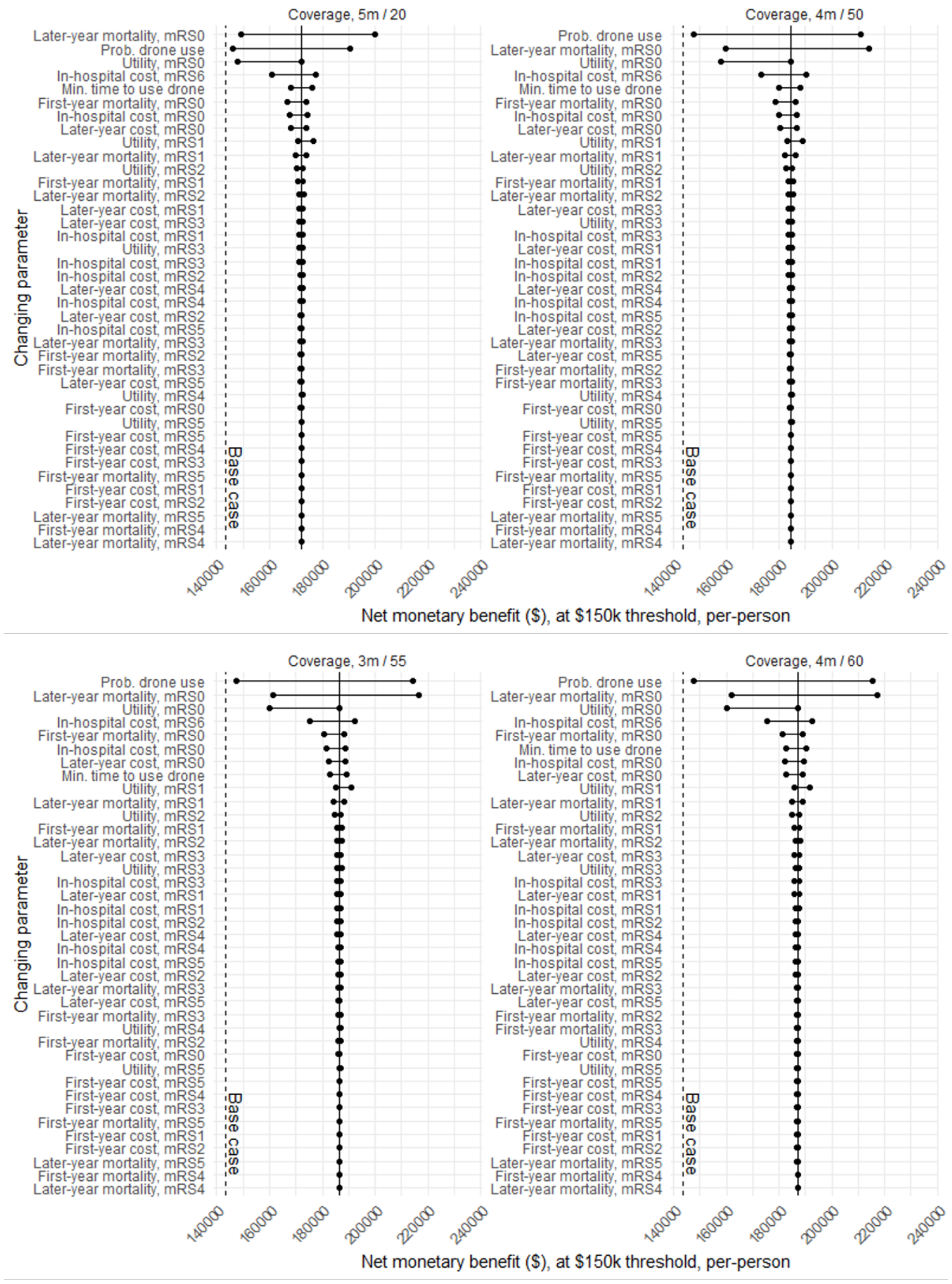
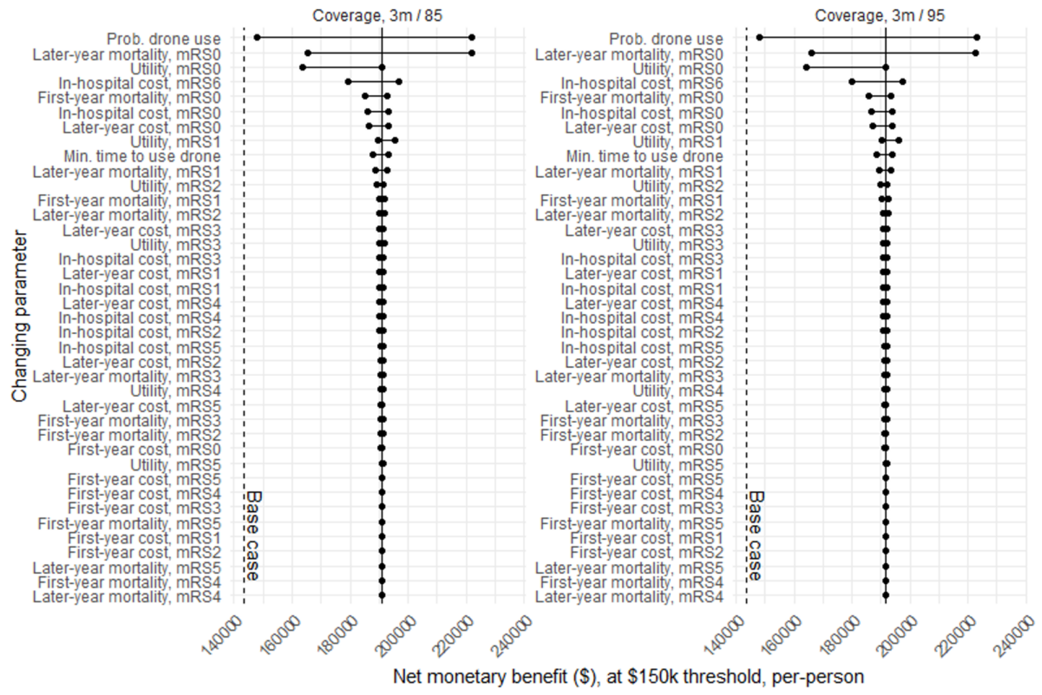
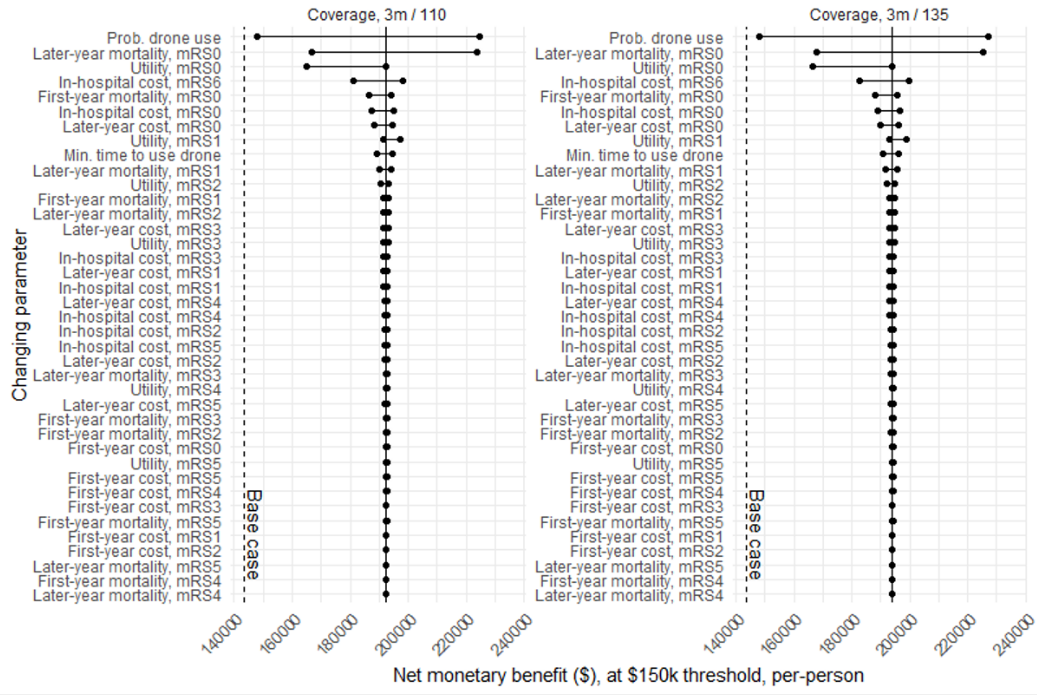
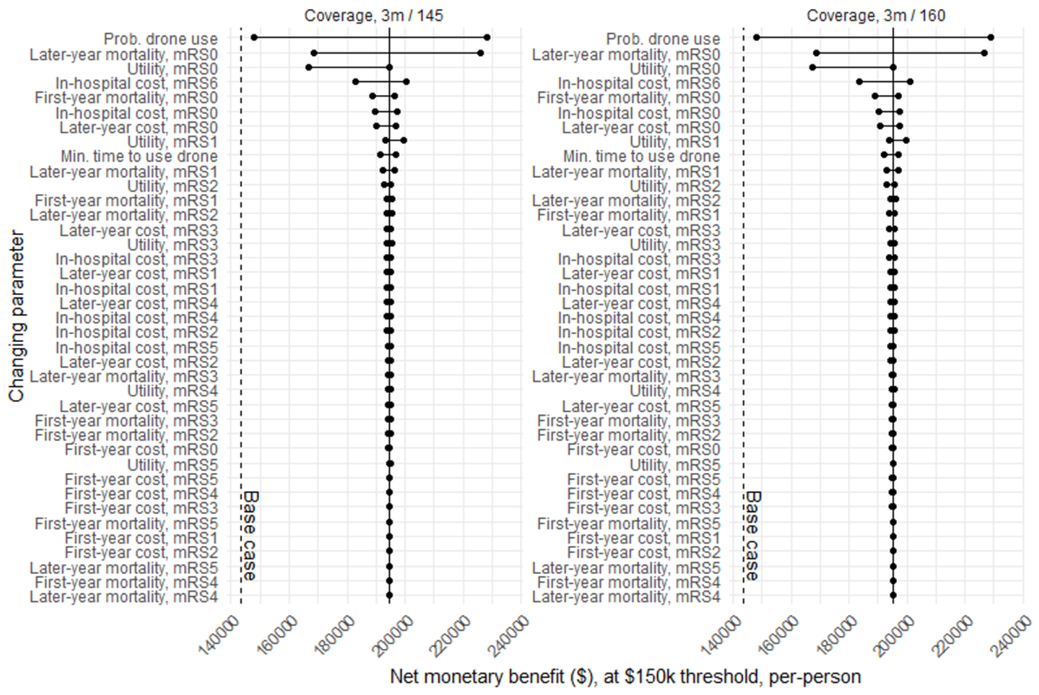


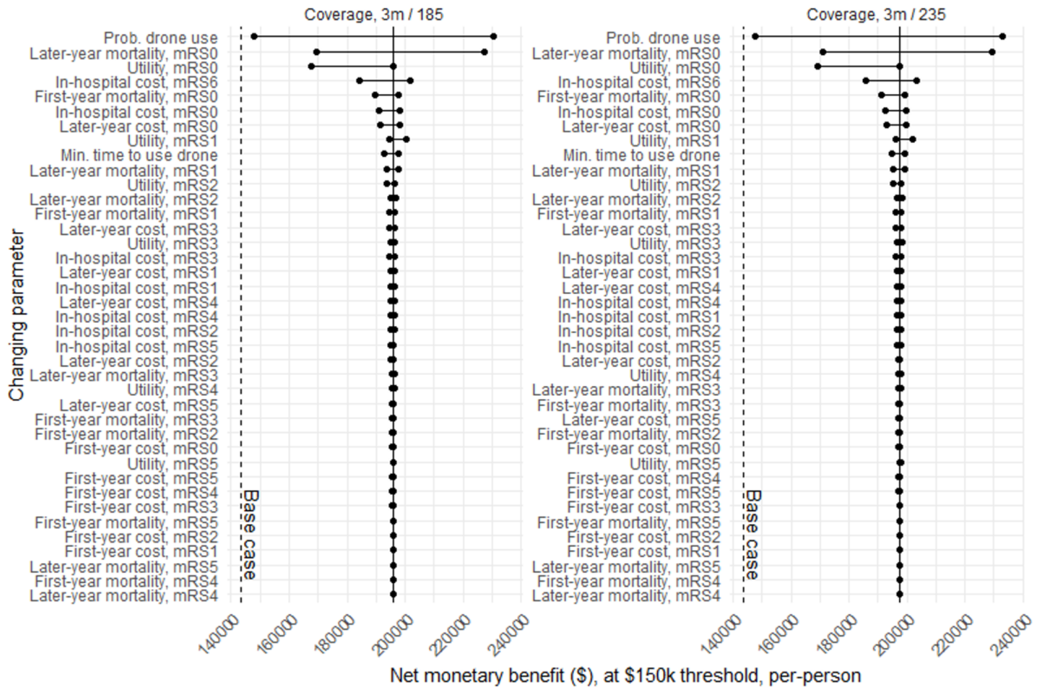
Figure A.7: Tornado plots showing sensitivity of net monetary benefit, at $150k threshold, to all model parameters, for all drone networks on the Pareto frontier. Solid line indicates NMB at the original parameter values. Drone network is labeled as: type of drone network / # drones. Parameters, listed on the x-axis, are ordered, for each drone network, in decreasing order of sensitivity

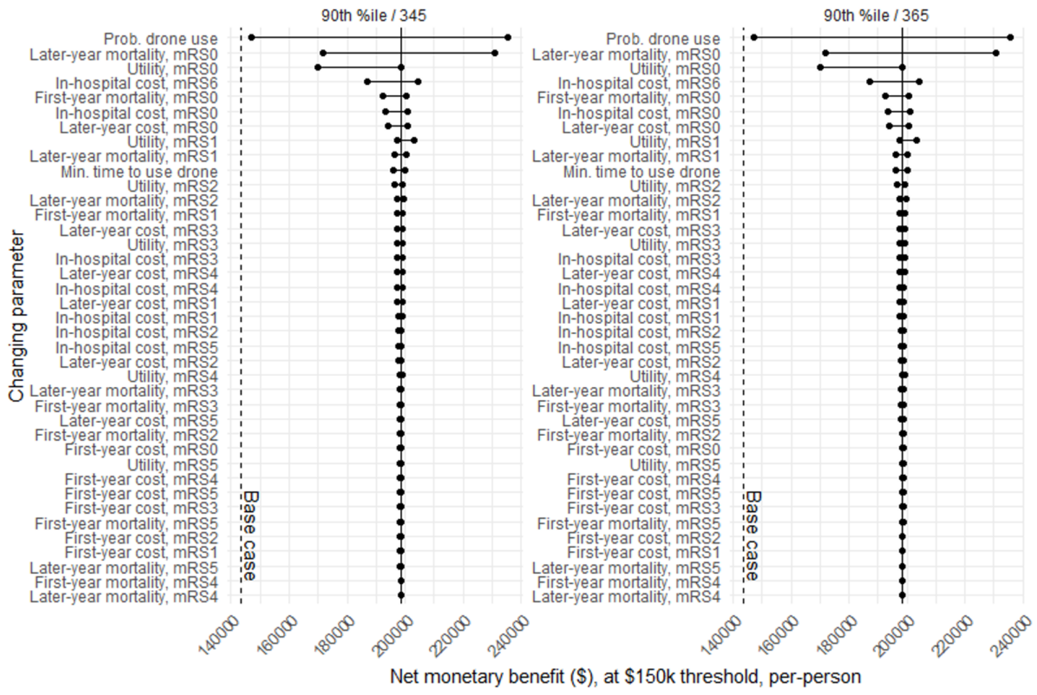


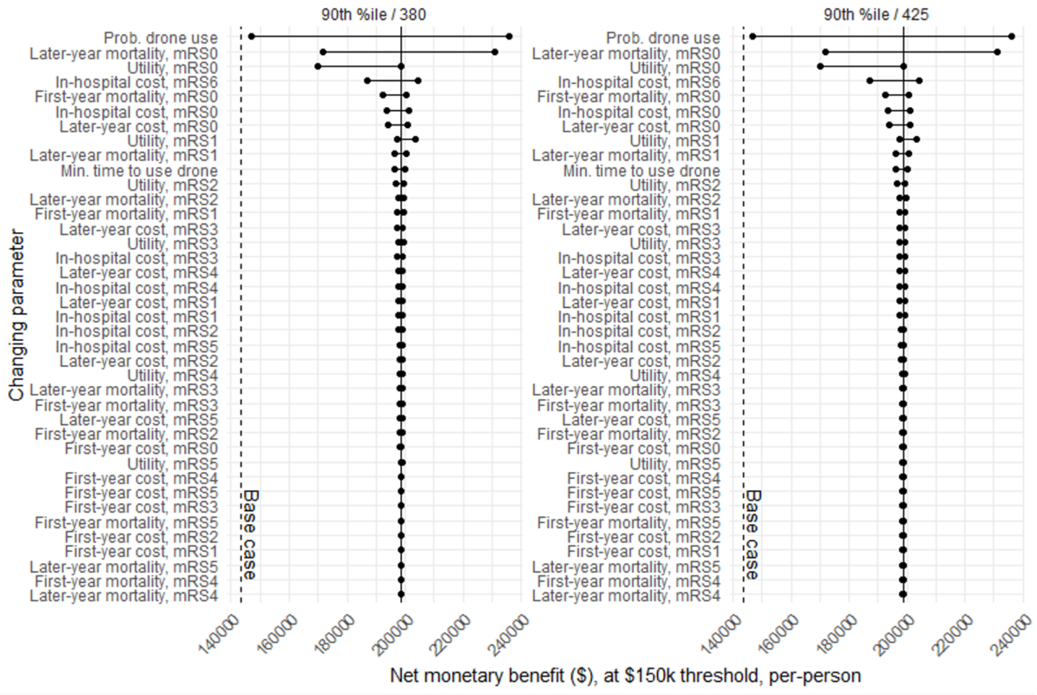

Coverage, 3m / 85
Coverage, 3m / 95
Changing parameter
Prob. drone use
Later-year mortality, mRS0
Utility, mRS0
In-hospital cost, mRS6
First-year mortality, mRS0
In-hospital cost, mRS0
Later-year cost, mRS0
Utility, mRS1
Min. time to use drone
Later-year mortality, mRS1
Utility, mRS2
First-year mortality, mRS1
Later-year mortality, mRS2
Later-year cost, mRS3
Utility, mRS3
In-hospital cost, mRS3
Later-year cost, mRS1
In-hospital cost, mRS1
Later-year cost, mRS4
In-hospital cost, mRS4
In-hospital cost, mRS2
In-hospital cost, mRS5
Later-year cost, mRS2
Later-year mortality, mRS3
Utility, mRS4
Later-year cost, mRS5
First-year mortality, mRS3
First-year mortality, mRS2
First-year cost, mRS0
Utility, mRS5
First-year cost, mRS5
First-year cost, mRS4
First-year cost, mRS3
First-year mortality, mRS5
First-year cost, mRS1
First-year cost, mRS2
Later-year mortality, mRS5
First-year mortality, mRS4
Later-year mortality, mRS4
Base case
140000
160000
180000
200000
220000
240000
Net monetary benefit ($), at $150k threshold, per-person

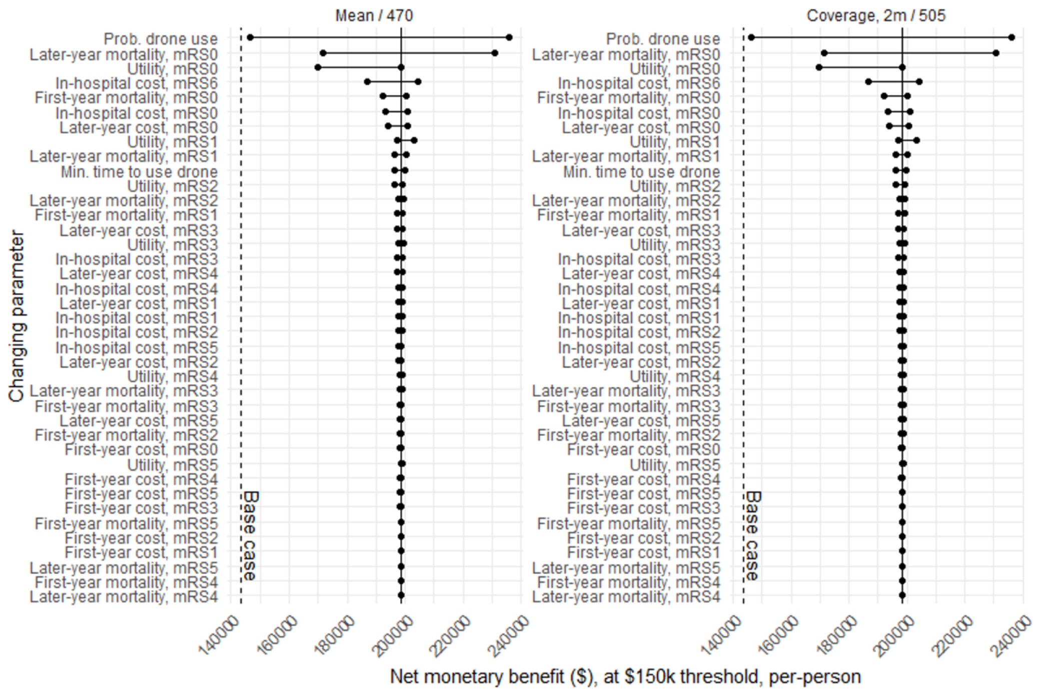

Coverage, 3m / 110
Coverage, 3m / 135
Changing parameter
Prob. drone use
Later-year mortality, mRS0
Utility, mRS0
In-hospital cost, mRS6
First-year mortality, mRS0
In-hospital cost, mRS0
Later-year cost, mRS0
Utility, mRS1
Min. time to use drone
Later-year mortality, mRS1
Utility, mRS2
First-year mortality, mRS1
Later-year mortality, mRS2
Later-year cost, mRS3
Utility, mRS3
In-hospital cost, mRS3
Later-year cost, mRS1
In-hospital cost, mRS1
Later-year cost, mRS4
In-hospital cost, mRS4
In-hospital cost, mRS2
In-hospital cost, mRS5
Later-year cost, mRS2
Later-year mortality, mRS3
Utility, mRS4
Later-year cost, mRS5
First-year mortality, mRS3
First-year mortality, mRS2
First-year cost, mRS0
Utility, mRS5
First-year cost, mRS5
First-year cost, mRS4
First-year cost, mRS3
First-year mortality, mRS5
First-year cost, mRS1
First-year cost, mRS2
Later-year mortality, mRS5
First-year mortality, mRS4
Later-year mortality, mRS4
Base case
140000
160000
180000
200000
220000
240000
Net monetary benefit ($), at $150k threshold, per-person


Coverage, 3m / 145
Coverage, 3m / 160
Changing parameter
Prob. drone use
Later-year mortality, mRS0
Utility, mRS0
In-hospital cost, mRS6
First-year mortality, mRS0
In-hospital cost, mRS0
Later-year cost, mRS0
Utility, mRS1
Min. time to use drone
Later-year mortality, mRS1
Utility, mRS2
First-year mortality, mRS1
Later-year mortality, mRS2
Later-year cost, mRS3
Utility, mRS3
In-hospital cost, mRS3
Later-year cost, mRS1
In-hospital cost, mRS1
Later-year cost, mRS4
In-hospital cost, mRS4
In-hospital cost, mRS2
In-hospital cost, mRS5
Later-year cost, mRS2
Later-year mortality, mRS3
Utility, mRS4
Later-year cost, mRS5
First-year mortality, mRS3
First-year mortality, mRS2
First-year cost, mRS0
Utility, mRS5
First-year cost, mRS5
First-year cost, mRS4
First-year cost, mRS3
First-year mortality, mRS5
First-year cost, mRS1
First-year cost, mRS2
Later-year mortality, mRS5
First-year mortality, mRS4
Later-year mortality, mRS4
Base case
140000
160000
180000
200000
220000
240000
Net monetary benefit ($), at $150k threshold, per-person


Coverage, 3m / 185
Coverage, 3m / 235
Changing parameter
Prob. drone use
Later-year mortality, mRS0
Utility, mRS0
In-hospital cost, mRS6
First-year mortality, mRS0
In-hospital cost, mRS0
Later-year cost, mRS0
Utility, mRS1
Min. time to use drone
Later-year mortality, mRS1
Utility, mRS2
Later-year mortality, mRS2
First-year mortality, mRS1
Later-year cost, mRS3
Utility, mRS3
In-hospital cost, mRS3
Later-year cost, mRS1
In-hospital cost, mRS1
Later-year cost, mRS4
In-hospital cost, mRS4
In-hospital cost, mRS2
In-hospital cost, mRS5
Later-year cost, mRS2
Later-year mortality, mRS3
Utility, mRS4
Later-year cost, mRS5
First-year mortality, mRS3
First-year mortality, mRS2
First-year cost, mRS0
Utility, mRS5
First-year cost, mRS5
First-year cost, mRS4
First-year cost, mRS3
First-year mortality, mRS5
First-year cost, mRS2
First-year cost, mRS1
Later-year mortality, mRS5
First-year mortality, mRS4
Later-year mortality, mRS4
Base case
140000
160000
180000
200000
220000
240000
Net monetary benefit ($), at $150k threshold, per-person


90th %ile / 345
90th %ile / 365
Changing parameter
Prob. drone use
Later-year mortality, mRS0
Utility, mRS0
In-hospital cost, mRS6
First-year mortality, mRS0
In-hospital cost, mRS0
Later-year cost, mRS0
Utility, mRS1
Later-year mortality, mRS1
Min. time to use drone
Utility, mRS2
Later-year mortality, mRS2
First-year mortality, mRS1
Later-year cost, mRS3
Utility, mRS3
In-hospital cost, mRS3
Later-year cost, mRS4
In-hospital cost, mRS4
Later-year cost, mRS1
In-hospital cost, mRS1
In-hospital cost, mRS2
In-hospital cost, mRS5
Later-year cost, mRS2
Utility, mRS4
Later-year mortality, mRS3
First-year mortality, mRS3
Later-year cost, mRS5
First-year mortality, mRS2
First-year cost, mRS0
Utility, mRS5
First-year cost, mRS4
First-year cost, mRS5
First-year cost, mRS3
First-year mortality, mRS5
First-year cost, mRS2
First-year cost, mRS1
Later-year mortality, mRS5
First-year mortality, mRS4
Later-year mortality, mRS4
Base case
140000
160000
180000
200000
220000
240000
Net monetary benefit ($), at $150k threshold, per-person


90th %ile / 380
90th %ile / 425
Changing parameter
Prob. drone use
Later-year mortality, mRS0
Utility, mRS0
In-hospital cost, mRS6
First-year mortality, mRS0
In-hospital cost, mRS0
Later-year cost, mRS0
Utility, mRS1
Later-year mortality, mRS1
Min. time to use drone
Utility, mRS2
Later-year mortality, mRS2
First-year mortality, mRS1
Later-year cost, mRS3
Utility, mRS3
In-hospital cost, mRS3
Later-year cost, mRS4
In-hospital cost, mRS4
Later-year cost, mRS1
In-hospital cost, mRS1
In-hospital cost, mRS2
In-hospital cost, mRS5
Later-year cost, mRS2
Utility, mRS4
Later-year mortality, mRS3
First-year mortality, mRS3
Later-year cost, mRS5
First-year mortality, mRS2
First-year cost, mRS0
Utility, mRS5
First-year cost, mRS4
First-year cost, mRS5
First-year cost, mRS3
First-year mortality, mRS5
First-year cost, mRS2
First-year cost, mRS1
Later-year mortality, mRS5
First-year mortality, mRS4
Later-year mortality, mRS4
Base case
140000
160000
180000
200000
220000
240000
Net monetary benefit ($), at $150k threshold, per-person


Mean / 470
Coverage, 2m / 505
Changing parameter
Prob. drone use
Later-year mortality, mRS0
Utility, mRS0
In-hospital cost, mRS6
First-year mortality, mRS0
In-hospital cost, mRS0
Later-year cost, mRS0
Utility, mRS1
Later-year mortality, mRS1
Min. time to use drone
Utility, mRS2
Later-year mortality, mRS2
First-year mortality, mRS1
Later-year cost, mRS3
Utility, mRS3
In-hospital cost, mRS3
Later-year cost, mRS4
In-hospital cost, mRS4
Later-year cost, mRS1
In-hospital cost, mRS1
In-hospital cost, mRS2
In-hospital cost, mRS5
Later-year cost, mRS2
Utility, mRS4
Later-year mortality, mRS3
First-year mortality, mRS3
Later-year cost, mRS5
First-year mortality, mRS2
First-year cost, mRS0
Utility, mRS5
First-year cost, mRS4
First-year cost, mRS5
First-year cost, mRS3
First-year mortality, mRS5
First-year cost, mRS2
First-year cost, mRS1
Later-year mortality, mRS5
First-year mortality, mRS4
Later-year mortality, mRS4
Base case
140000
160000
180000
200000
220000
240000
Net monetary benefit ($), at $150k threshold, per-person

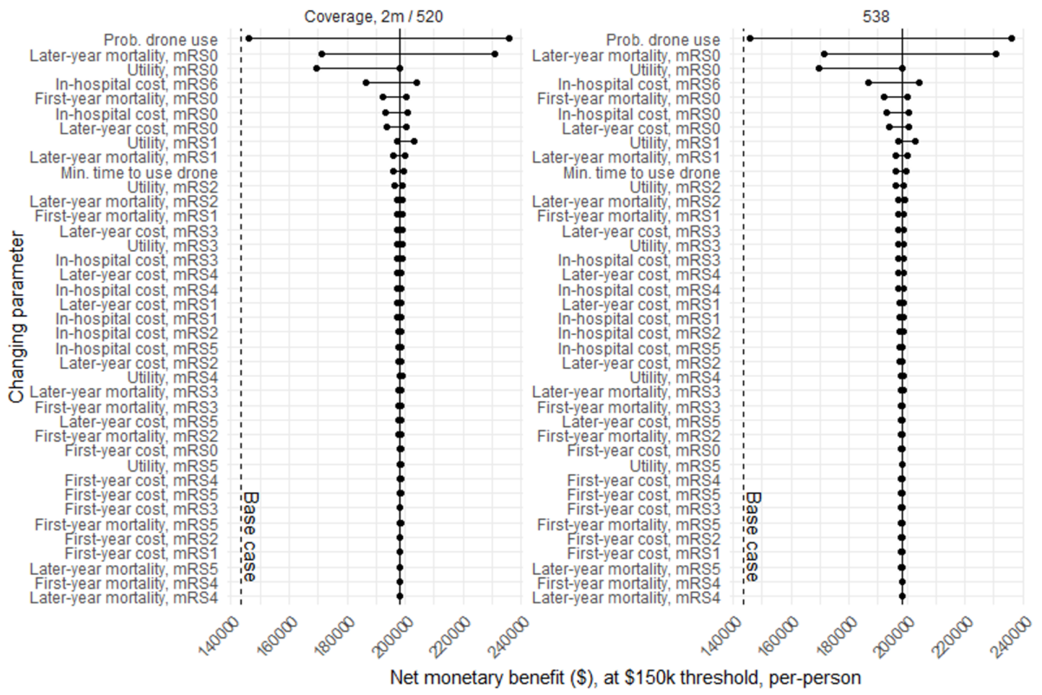

Coverage, 2m / 520
538
Changing parameter
Prob. drone use
Later-year mortality, mRS0
Utility, mRS0
In-hospital cost, mRS6
First-year mortality, mRS0
In-hospital cost, mRS0
Later-year cost, mRS0
Utility, mRS1
Later-year mortality, mRS1
Min. time to use drone
Utility, mRS2
Later-year mortality, mRS2
First-year mortality, mRS1
Later-year cost, mRS3
Utility, mRS3
In-hospital cost, mRS3
Later-year cost, mRS4
In-hospital cost, mRS4
Later-year cost, mRS1
In-hospital cost, mRS1
In-hospital cost, mRS2
In-hospital cost, mRS5
Later-year cost, mRS2
Utility, mRS4
Later-year mortality, mRS3
First-year mortality, mRS3
Later-year cost, mRS5
First-year mortality, mRS2
First-year cost, mRS0
Utility, mRS5
First-year cost, mRS4
First-year cost, mRS5
First-year cost, mRS3
First-year mortality, mRS5
First-year cost, mRS2
First-year cost, mRS1
Later-year mortality, mRS5
First-year mortality, mRS4
Later-year mortality, mRS4
Base case
140000
160000
180000
200000
220000
240000
Net monetary benefit ($), at $150k threshold, per-person

Figure A.8: Net monetary benefits of all networks, at thresholds of \$50k, \$100k, \$150k (labels across top), for the scenarios where drone operational costs are 1x (i.e., standard), 2x, and 5x (labels on right-hand side) of the values reported in Table 1

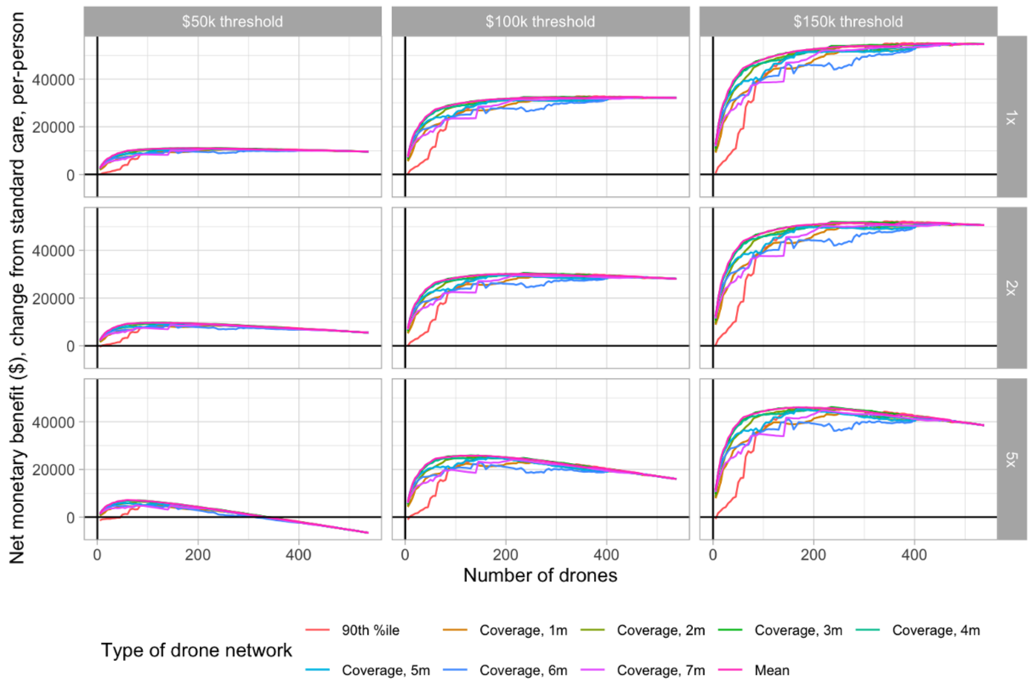


Figure A.9: Breakeven multiplier of the drone cost of each drone network, at thresholds of \$50k, \$100k, \$150k. Values represent how much drone costs would need to be multiplied so that the drone network has the same NMB as standard care

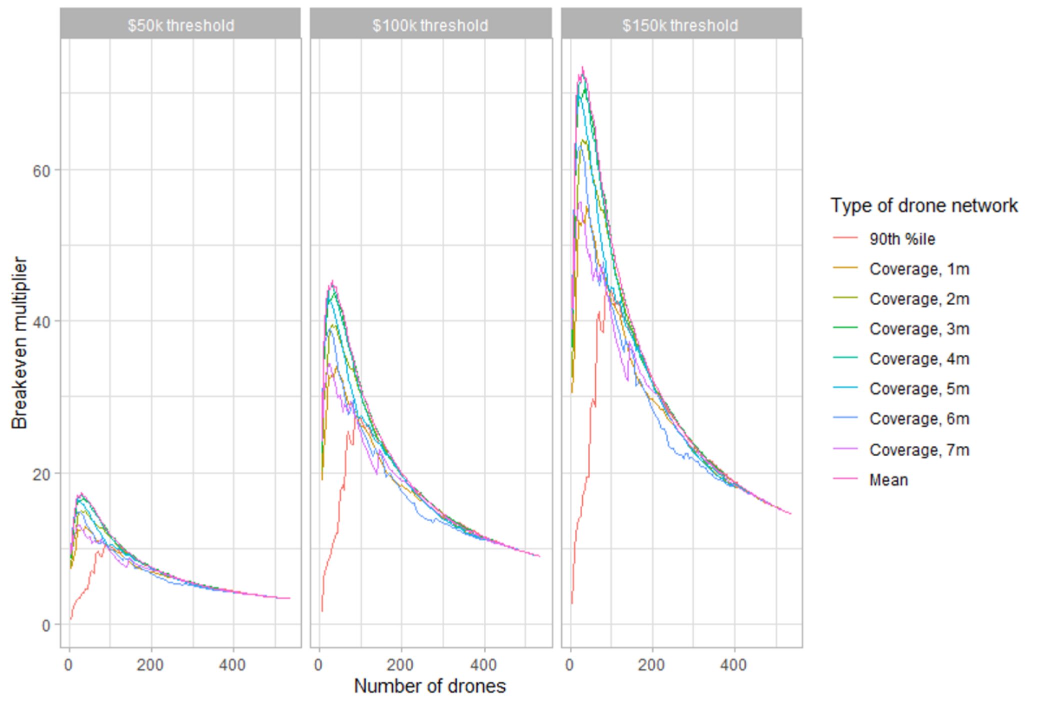


Figure A.10: Net monetary benefits of all networks, at $150k threshold, when randomly excluding 10%, 25%, 50%, and 75% of patients from being dispatched a drone. 100 draws were performed for each exclusion percentage, and metrics averaged across draws

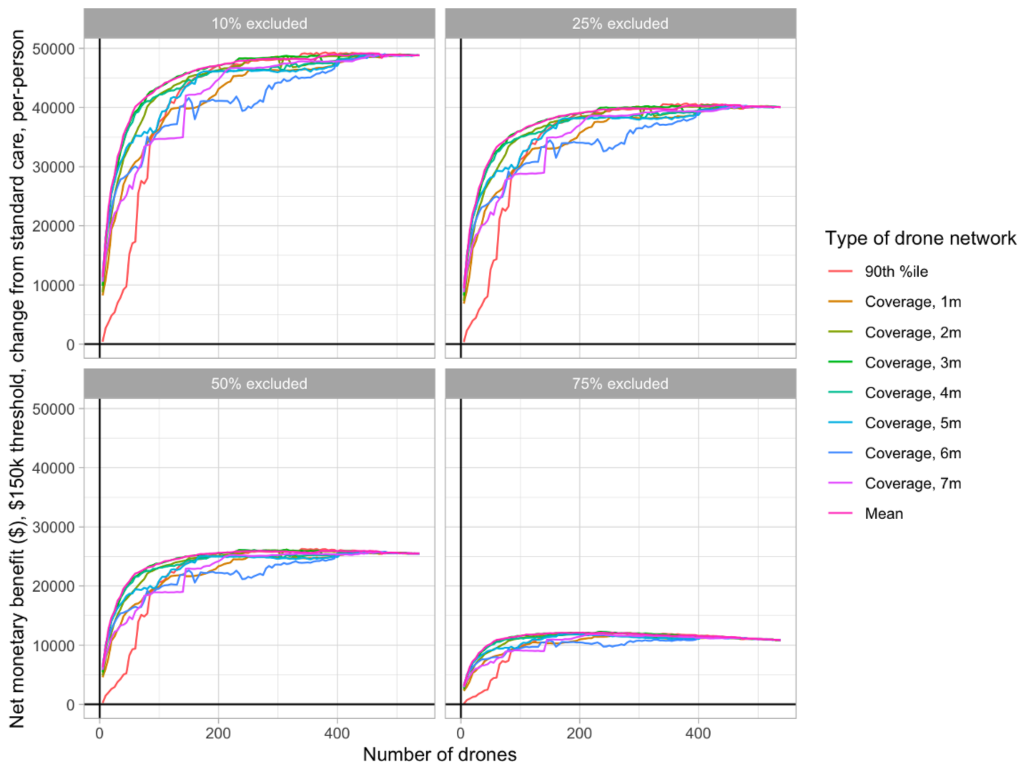


Figure A.11: Net monetary benefits of all networks, at thresholds of \$50k, \$100k, \$150k (labels across top), when excluding the n=1006 patients who survived hospital discharge but had a missing mRS

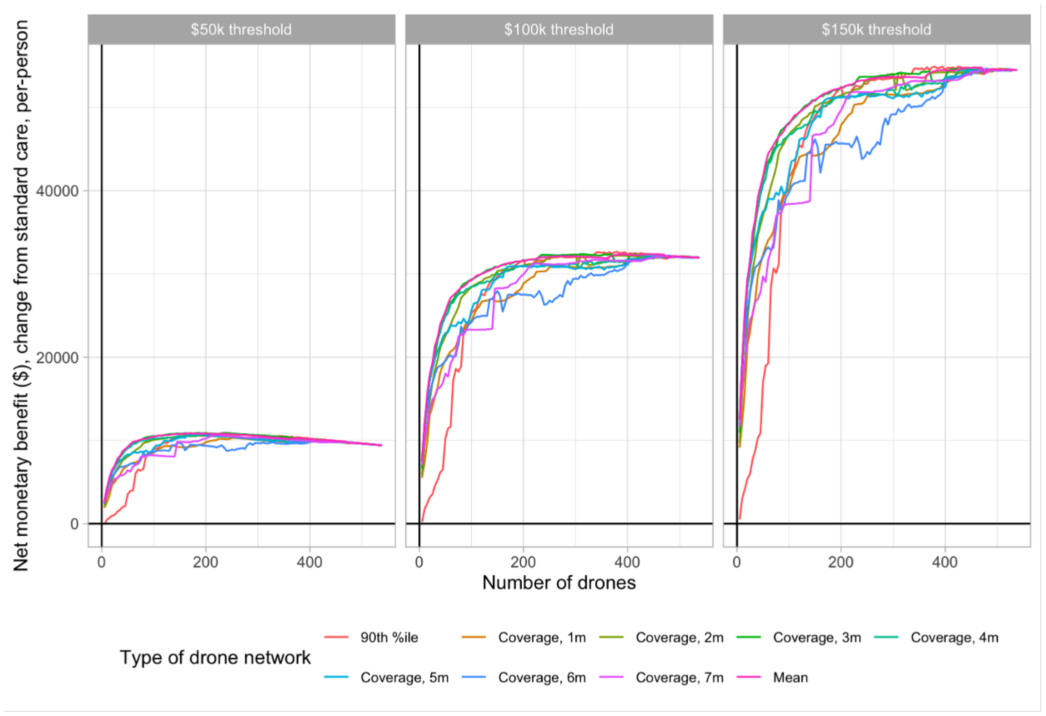
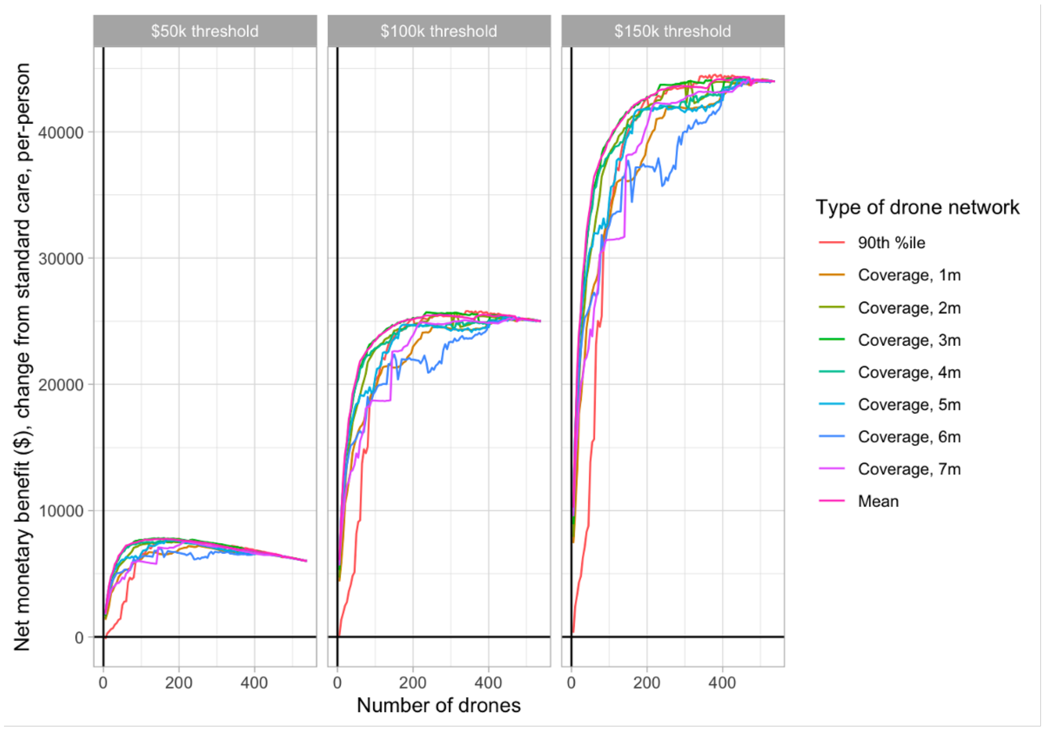


Figure A.12: Net monetary benefits of all networks, at thresholds of \$50k, \$100k, \$150k (labels across top), when scaling down all utility values to account for Canadian population baseline utilities


$50k threshold
$100k threshold
$150k threshold
Net monetary benefit ($), change from standard care, per-person
Number of drones
Type of drone network
90th %ile
Coverage, 1m
Coverage, 2m
Coverage, 3m
Coverage, 4m
Coverage, 5m
Coverage, 6m
Coverage, 7m
Mean