Resource Allocation for Constrained Multi-Agent Systems

Anna Gautier

Wolfson College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

April 21, 2023

Acknowledgements

Most people get one supervisor, I was lucky enough to have two wonderful supervisors. Nick and Mike, I am immensely grateful for not just your academic support, but also your moral and pastoral support, particularly during 2020. Thank you for continuing to believe in me. A very special thanks to Bruno, who was not my supervisor on paper, but has been an indispensable mentor throughout my DPhil. This thesis would not exist without you. Thank you to Ani and Matthijs for examining my thesis. I greatly appreciate the time and effort you put into my examination, and it was a pleasure to discuss my work with you both. Wendy, thank you for constantly supporting me. Joining AIMS was the best choice I could have made. Thank you to AIMS and AWS for providing the funding for my DPhil. I would also like to express my gratitude to the amazing educators who taught me in both grade school at Great Valley and in my undergraduate at WashU, all of my whom helped foster my love of learning and my interest in science and mathematics. In particular, I'd like to thank Mrs. Rothberg, who encouraged me to read and to learn about the world. It's no accident that most of our book club ended up in grad school.

My five years in Oxford has been filled with friendship. With my two supervisors, I gained two research groups full of lifelong friends. Between board game nights and jetting off to Austria, Italy, and Spain (for work!) I could not ask for better peers. Thank you to Gianmarco, Andres, and the rest of my OUBTS family for helping me distract myself with my favourite hobby. Appearing on HBO is a close second to finishing a DPhil. I also have an immense amount of love in particular for four fellow women in STEM: Clarissa, Laura, Amanda, and Aletta: I'm frankly not sure how I survived at Oxford until you all came into my life. Speaking of female friendships, thank you to the friends who stuck with me across an ocean: Zoe (for every late night FaceTime), Maxine (for always brightening my day), Steph (for all our adventures), Mary-Brent (for listening to me talk about every and nothing), Natalie (for always understanding me), and Laura and Michelle (for forever feeling like home).

Finally, thank you so much to my parents for supporting me in my dreams, even when those dreams included a move half way across the world. I cannot express how much I appreciate everything you sacrificed to get me this far. I would not be here with out your love and support, and for that I am eternally grateful.

Abstract

In this thesis, we explore the challenges of resource allocation in multi-agent systems. In particular, we consider resource allocation for multi-agent planning, where agents make a series of independent decisions to achieve their own goals. We focus on three main challenges: uncertain domains, differing resource types, and non-cooperative settings. In uncertain domains, stochasticity in the environment leads to agents who are uncertain about their future resource use, which can present challenges for a resource allocation mechanism. Different resource types have different challenges, and increasing the types of resources being allocated in one problem increases the space of possible resource allocations which must be searched. Finally, in non-cooperative settings, agents may compete for resources, and competition may lead to agents lying about their preferences. Lying can, in turn, cause challenges for a mechanism that seeks to optimise a global property.

We address these three challenges through the lens of two classical planning challenges, Multi-Agent Markov Decision Processes and Multi-Agent Path Finding. In the context of weakly-coupled Multi-Agent Markov Decision Processes, we consider the problem of chance-constrained resource allocation and present an auction based method which is applicable to non-cooperative settings. Next, we consider the problem of risk-constrained resource allocation in cooperative Multi-Agent Markov Decision Processes. Finally, we consider the problem of allocating many differing resources in non-cooperative, deterministic Multi-Agent Pathfinding settings. In all settings, we experimentally evaluate the performance of our methods compared to state-of-the-art techniques.

Contents

List of Figures

List of Tables

List of Abbreviations

ACCR . . . . . Auction for Chance-Constrained Resources

CA . . . . . Combinatorial Auction

CCMDP . . . Chance-Constrained Markov Decision Process

CCMMDP . . Chance-Constrained Multi-Agent Markov Decision Process

CMDP . . . . Constrained Markov Decision Process

CMMDP . . . Constrained Multi-Agent Markov Decision Process

CVaR . . . . . Conditional Value-At-Risk

ILP . . . . . Integer Linear Program

INLP . . . . . Integer Nonlinear Program

LTL . . . . . Linear Temporal Logic

LP . . . . . Linear Program

MAPF . . . . . Multi-Agent Pathfinding

MDP . . . . . Markov Decision Process

MMDP . . . . Multi-Agent Markov Decision Process

PKA . . . . . Privileged Knowledge Auction

POMDP . . . Partially Observable Markov Decision Process

RCA . . . . . Risk Contribution Approach

RCMDP . . . Risk-Constrained Markov Decision Process

RCMMDP . . Risk-Constrained Multi-Agent Markov Decision Process

SSI . . . . . Sequential Single Item (auction)

VaR . . . . . Value-At-Risk

VCG . . . . . Vickrey-Clarke-Groves

DTMC . . . . Discrete Time Markov Chain

List of Mathematical Notation

General Terms

h . . . . . A finite time horizon.

k . . . . . A number of resources that is $\leq L$.

L . . . . . A finite number of resources.

m . . . . . A finite number of bids.

n . . . . . A finite number of agents.

γ . . . . . A stepsize.

Decision Making Under Uncertainty

A_i . . . . . A set of single-agent actions.

A . . . . . A set of multi-agent actions.

C_i . . . . . A single-agent cost function.

C . . . . . A multi-agent cost function.

$\mathcal{C}_{i,\pi_i}$ . . . . . A single-agent induced-cost random variable.

$\mathcal{C}_{\pi}$ . . . . . A multi-agent induced-cost random variable.

$E_{\pi}[Y]$ . . . . . The expected value of random variable Y under policy π.

$F(z)$ . . . . . A cumulative distribution function.

$\mathcal{N}(\mu, \sigma^2)$ . . . . A normal distribution with mean μ and standard deviation σ.

$\mathcal{M}_i$ . . . . . A single-agent MDP.

$\mathcal{M}$ . . . . . A multi-agent MDP.

$P_{\pi}[D]$ . . . . . The probability of event D under policy π.

R_i . . . . . A single-agent reward function.

R . . . . . A multi-agent reward function.

$\mathcal{R}_{i,\pi_i}$ . . . . . A single-agent induced-reward random variable.

$\mathcal{R}_{\pi}$ . . . . . A multi-agent induced-reward random variable.

S_i . . . . . A set of single-agent states.

S . . . . . A set of multi-agent states.

T_i . . . . . A single-agent transition function.

T . . . . . A multi-agent transition function.

δ . . . . . A confidence level.

$\epsilon_{i,k}$ . . . . . A single-agent probability of exceeding k resources.

π_i . . . . . A single-agent policy.

π . . . . . A multi-agent policy.

Mechanism Design

$b_{i,k}$ $(b_{i,\alpha})$ . . . . A single-agent bid for k (the set of α) resources.

$\mathcal{F}$ . . . . . An allocation of resources to agents.

$\mathcal{F}^{-i}$ . . . . . An allocation of resources to agents, *excluding* agent i.

p_i . . . . . A single-agent price.

v_i . . . . . A single-agent valuation function.

$\mathcal{Z}$ . . . . . A set of items.

Γ . . . . . The set of feasible allocations.

σ_i . . . . . A single-agent functions mapping valuations to bids.

Ω . . . . . A set of outcomes.

Multi-Agent Pathfinding

$\mathcal{G}$ . . . . . A graph.

g_i . . . . . A single-agent goal vertex

E . . . . . A set of edges.

V . . . . . A set of vertices.

V^* . . . . . The set of valid paths.

r_i . . . . . A single-agent starting vertex.

w_i . . . . . A valid path.

x_d . . . . . A vertex.

1 Introduction

This thesis explores the problem of resource allocation in multi-agent systems. In particular, we focus on the area of planning, where independent agents need to make a sequence of decisions in order to achieve some goal. Our focus is on two classic planning challenges: Markov Decision Processes [Bellman, 1957, Puterman, 2014] and Multi-Agent Pathfinding [Stern et al., 2019]. Often, the result of decisions in both domains requires some sort of resource, whether it be electricity, money, or the use of a physical object. When multiple agents make decisions in the same system, they are therefore coupled by the scarce resources that they need to share, even when their operations are otherwise independent, as in [Meuleau et al., 1998]. We approach this resource allocation problem from the perspective of a system designer who is interested in creating a mechanism to preallocate resources among agents. This allows agents to make more informed decisions based on their allocated number of resources. We argue that there are three major challenges that face resource allocation in multi-agent planning: uncertainty, differing resource types, and non-cooperative settings.

Allocating Resources Under Uncertainty

Uncertain domains arise when agents cannot deterministically model the environment they interact in. Multi-agent systems with uncertainty include

unmanned aerial vehicles [Yun et al., 2022], autonomous driving [Yu et al., 2019], and human-AI collaboration [Ramchurn et al., 2016]. We focus on the uncertainty that arises from inherent stochasticity in the environment, modelled as Multi-Agent Markov Decision Processes (MMDPs) [Boutilier, 1996, Puterman, 2014]. In multi-agent resource allocation problems, uncertainty arises in wireless networks [Feriani and Hossain, 2021], budget allocation [Boutilier and Lu, 2016], and power consumption [de Nijs et al., 2019]. But in resource allocation problems, the uncertainty in the environment leads to uncertainty over resource usage. Imagine a robot that is located at the bottom of a hill and needs to reach the top.

Example 1. *Suppose that we can model the robot's ability to ascend the hill: it uses one unit of energy and has a 90% success rate of reaching the top and a 10% chance of rolling back down. Suppose the robot decides to always ascend the hill when they are at the bottom. The robot may get lucky and be in the 90% of the distribution where they reach the top on their first try and consume only one unit of the resource. Or the robot may get unlucky and be in the .1% of the distribution where it takes at least 4 tries and thus they need to consume at least five units of the resource.*

Even a single agent with a simple, deterministic decision making process of "always ascend the hill when you are at the bottom" will have an uncertain resource usage due to the aleatoric uncertainty in the environment. The distributions over costs get increasingly more complex with larger environments and more agents, which makes resource uncertainty in sequential decision making a difficult problem. It is also not obvious how to preallocate resources when agents are uncertain; many problem variants exist in multi-agent planning including Wu and Durfee [2010], Yost and Washburn [2000], de Nijs et al. [2017], each of which has different pros and cons that we explore throughout the thesis.

Allocating Different Resource Types

Unsurprisingly, the methods for allocating different categories of resources vary. We exclusively consider finite resources because scarcity is a prerequisite for allocation problems. We first consider the simplest type of resource allocation problem, where there is a discrete, finite number of units of a homogenous resource. This is known as a multi-unit resource [Kwasnica and Sherstyuk, 2013]. In this case, agents have preferences over how many units of each resource they are allocated, which are monotonic. Despite being simple, this category of allocation problem covers a wide range of applications, including energy markets [Wolfram, 1997] and government security markets [Ausubel and Cramton, 1998]. Another category of resources is a continuous, finite homogeneous resource, or equivalently a single, infinitely divisible unit of a resource [Maheswaran and Başar, 2003]. Another category is a single resource that is time-dependant and renewable, known as a instantaneous resource [de Nijs et al., 2021]. In this case, agents' preferences at one point in time might depend on what they are allocated at another point in time. This interdependence adds complexity to how agents can calculate and express their preferences. Similarly, if there are many different, finite, but non-homogeneous resources, agents' preferences may depend on being allocated them jointly. In these settings, we can make far fewer a priori assumptions on agent preferences, because agent preferences may be highly dependent on the interaction between different resources. For example, a hammer and a nail may only be useful if you can use both at the same time. When all the resources are unique, there are more possible outcomes to consider. More categories of resources can represent more interesting problems, but result in a combinatorial problem which comes with a computational hurdle [Cramton et al., 2006].

Allocating Resources in Non-Cooperative Settings

The final challenge we consider is the non-cooperative setting. As more decision making becomes autonomous, more unaligned autonomous agents will begin to interact with each other. Consider, for example, autonomous robotics. As autonomous robot systems are deployed at a large scale in both public and private spaces, robots owned

and operated by competing organisations will be required to interact. Autonomous robots have already appeared in grocery stores [Bogue, 2019, Begley et al., 2020], in hospitals [Khan et al., 2020, Kyrarini et al., 2021], and on our neighbourhood sidewalks [Jennings and Figliozzi, 2019]. To date, only a few companies design these robots, but imagine a future where every company delivers their packages with a sidewalk robot. How do those robots share the sidewalk? How do grocery store robots with different tasks divide time connecting to chargers? How do hospital robots from different wards share bandwidth on Wi-Fi networks? These questions are resource allocation questions, but they are also *non-cooperative* scenarios. We specifically consider weakly-coupled, non-cooperative systems, where agents' goals do not depend on the actions of the other agents. This is distinct from adversarial settings, or more general forms of games. In these settings, agents are coupled only through their shared resource. In weakly-coupled systems, agent planning can be partially de-coupled through novel single-agent planning and intelligent search methods, e.g., [Yost and Washburn, 2000, de Nijs et al., 2017]. But, when agents with different goals interact and compete for resources, it is also vital to design systems that can anticipate and plan for this competition.

1.1 Contributions

In this thesis, we address the problem of resource allocation under uncertainty on the three axes described above: uncertain domains, differing resource types, and non-cooperative scenarios. The thesis chapters, and how they align with each axis, are represented in Table 1.1.

Additionally, we summarise the main questions that are answered by our work.

Q1 - Problem Definition: Can we define new classes of multi-agent resource allocation problems, specifically in planning in multi-agent Markov decision processes?

Table 1.1: A description of each chapter's relevance to the three main challenges of multi-agent decision making.

	Chapter 4	Chapter 5	Chapter 6
Uncertain Domain?	Yes	Yes	No
Rich Uncertainty?	No	Yes	No
Multiple Resource Types?	No	No	Yes
Non-Cooperative?	Yes	No	Yes

Chapter 4: Allocating Chance-Constrained Resource
Chapter 5: Allocating Risk-Constrained Resources
Chapter 6 : Allocating Space: A Non-Cooperative Approach to MAPF

Q2 - Problem Extension: Can we extend already existing multi-agent resource allocation problems to be applicable to more scenarios, in multi-agent Markov decision processes and multi-agent pathfinding?

Q3 - Novel Solutions: Can we find unique solution methods to the problems defined in Q1?

Q4 - Algorithmic Improvement: Can we find unique solution methods to the problems defined in Q3?

Q5 - Decentralised Computation: Can we design methodology in Q3 and Q4 that decentralises the computation of multi-agent planning through novel single-agent planning methods?

With respect to these questions, our main results are as follows.

First, we present a new non-cooperative auction mechanism called the Auction for Chance-Constrained Resources (ACCR). ACCR preallocates multi-unit resources under uncertainty among agents while limiting the chance of resource violations. We formally extend the already existing problem of allocating chance-constrained resources to the non-cooperative domain (**Q2**). We also present an algorithm that allows agents to generate a diverse set of bids via multi-objective reasoning, which are then submitted to the auction (**Q5**). Finally, we demonstrate empirically that our auction outperforms state-of-the-art cooperative techniques for chance-constrained multi-agent resource allocation in complex settings with up to hundreds of agents (**Q4**).

Second, we present a novel formulation of the risk-constrained multi-agent resource allocation problem, where we focus on optimising for reward while constraining the Conditional Value-at-Risk (CVaR) of the shared resource (**Q1**). CVaR is a measure of risk which requires deeper distributional information on agent's potential resource usage returns, in particular in the worst cases. While CVaR is well-studied in the single-agent setting, we consider the challenges that arise from the state and action space explosion in the multi-agent setting. In particular, we exploit risk contributions, a measure introduced in finance research which quantifies how much individual agents affect the joint risk. Using this notion, we define the Risk Contribution Algorithm (RCA) to solve this problem (**Q3**). Finally, we present a single-agent algorithm for estimating risk contributions (**Q5**).

Third, we consider the problem of allocating physical space in a graph in non-cooperative settings. We present the novel Privileged Knowledge Auction (PKA) to improve upon the already existing non-cooperative multi-agent pathfinding solutions (**Q4**). We also define a novel metric to generate sufficiently dissimilar single-agent plans (**Q5**) which allows for more problem settings to be solved.

1.1.1 Thesis Structure and Publications

In Chapter 2, we review the technical knowledge required for decision making under uncertainty and mechanism design, the two fields of research that span multiple chapters in the thesis. In Chapter 3, we discuss existing literature in the same categories. In Chapter 4, Allocating Chance-Constrained Resource, we introduce and evaluate our method for chance-constrained non-cooperative multi-agent resource allocation under uncertainty. Chapter 4 was originally published in Gautier et al. [2023a]. In Chapter 5, Allocating Risk-Constrained Resources, we introduce our method for risk-constrained cooperative multi-agent resource allocation under uncertainty. Chapter 5 was originally published in Gautier et al. [2023b]. In Chapter 6, Allocating Space: A Non-Cooperative Approach to MAPF, we introduce our method from allocating space as a resource in non-cooperative

multi-agent pathfinding. Chapter 6 was originally published in Gautier et al. [2022]. Finally, Chapter 7 summarises our important results.

Publications in Chronological Order

- [Gautier et al., 2022] Anna Gautier, Alex Stephens, Bruno Lacerda, Nick Hawes, and Michael Wooldridge. Negotiated path planning for non-cooperative multi-robot systems. In Proceedings of the Twenty-First International Conference on Autonomous Agents and Multiagent Systems, 2022.

- [Gautier et al., 2023a] Anna Gautier, Bruno Lacerda, Nick Hawes, and Michael Wooldridge. Multi-unit auctions for allocating chance-constrained resources. In Proceedings of the Thirty- Seventh AAAI Conference on Artificial Intelligence, 2023.

- [Gautier et al., 2023b] Anna Gautier, Marc Rigter, Bruno Lacerda, Nick Hawes, and Michael Wooldridge. Risk-constrained planning for multi-agent systems with shared resources. In Proceedings of the Twenty-Second International Conference on Autonomous Agents and Multiagent Systems, 2023.

2 Background

2.1 Decision Making Under Uncertainty

2.1.1 Markov Decision Processes

Markov Decision Processes (MDPs) are a common model for *single-agent* planning and decision making under uncertainty [Bellman, 1957, Puterman, 2014]. An agent models the current state of the environment and reasons over how their actions will affect the next state of the environment. A key feature of MDPs is that the outcomes of actions are stochastic. The problem of policy synthesis in MDPs considers how an agent should act given a) the current state of the environment and b) their future cumulative expected reward. Agents must reason over not only the reward they gain from their current action, but also how their current action will affect the environment and therefore their future ability to gain more reward.

Definition 1 (Markov Decision Process [Puterman, 2014])**.** *A (finite-horizon) MDP for a single agent i is defined by $\mathcal{M}_i = \langle S_i, A_i, T_i, R_i, h \rangle$, where S_i is the agent's state space, A_i is the agent's action set, and $T_i : S_i \times A_i \times S_i \to [0, 1]$ is the agent's transition function. Agents also have a reward function $R_i : S_i \times A_i \to \mathbb{R}$. $h \in \mathbb{N}^+$ defines the time horizon.*

An agent's goal is to synthesize a policy π_i, which can be *discrete* or *stochastic* and *stationary* or *non-stationary*.

Definition 2 (Deterministic Policy [Puterman, 2014]). *A deterministic (stationary) policy π_i maps from the state space to the set of actions. π_i is defined by $\pi : S_i \to A_i$ with $\pi(s_i) = a_i$ meaning that agent i should take action a at when in state s_i.*

Definition 3 (Stochastic (a.k.a. randomised) Policy [Puterman, 2014]). *A stochastic (stationary) policy π_i maps from the state space to the set of actions. π_i is defined by $\pi : S_i \to \Delta A_i$ with $\pi(s_i) = (prob_{a_1}, prob_{a_2}, ...)$, where ΔA_i is the set of probability distributions over elements in A_i. This means agent i should take action a_1 with probability $prob_{a_1}$ at when in state s_i.*

Definition 4 (Non-Stationary Policy [Puterman, 2014]). *A non-stationary (deterministic) policy π_i maps from the state space and current timestep to the set of actions. π_i is is defined by $\pi_i : S \times [h] \to A_i$ with $\pi_i(s_i, t) = a$ meaning that agent i should take action a at when in state s_i at timestep t.*

Even when policies are deterministic, i.e stationary, the outcome of executing as policy over an MDP is stochastic, as seen in Example 1, where an agent is trying to climb a hill. So we denote the probability of an event D under π_i as $P_{\pi_i}[D]$ and the expectation of a random variable Y under π_i as $E_{\pi_i}[Y]$.

Given an MDP, there are a variety of ways to synthesise optimal policies with respect to the reward function R. Classically, agents optimise for their cumulative reward function. Agents can optimise for either *discounted* or *undiscounted* reward.

Definition 5 (Cumulative Discounted Reward [Puterman, 2014]). *Given a discount factor $0 \leq \gamma < 1$, the cumulative discounted reward is defined by $E_{\pi_i}[\sum_{t=1}^{h} \gamma^t R(s_t, \pi_i(s_t))]$.*

The smaller the discount factor, the more the near-term is valued over the long-term. When $\gamma \to 1$, the present and the future approach being equally weighted. Note that s_t is a random variable. Returning to Example 1, there are two states: at the bottom of the hill and at the top of the hill. If s_1 is at the bottom of the hill, and $\pi_i(s_1)$ is to climb the hill, then s_2 depends on the success probability of climbing the hill, and is thus a random variable.

Definition 6 (Cumulative Undiscounted Reward [Puterman, 2014])**.** *The cumulative undiscounted reward is defined by* $E_{\pi_i}[\sum_{t=1}^{h} R(s_t, \pi_i(s_t))]$.

Because we focus on finite-horizon MDPs, we will exclusively consider cumulative undiscounted reward. For notational convenience we can define the random variable $\mathcal{R}_{i,\pi_i}$ to describe the cumulative reward agent i receives when executing policy π_i over the entire time horizon h. With this framing, the cumulative undiscounted reward is defined by $E_{\pi_i}[\mathcal{R}_{i,\pi_i}]$.

2.1.2 Markov Decision Processes with Constraints

The class of MDPs with constraints extends MDPs such that actions also incur a cost that corresponds to the consumption of a resource [Altman, 1999]. In an MDP with Constraint, the agent still wants to maximise their future expected reward, but they also need to constrain their resource usage.

Definition 7 (A Markov Decision Process with Constraint[1] [Puterman, 2014])**.** *A (finite-horizon) MDP with Constraint for agent i is defined by* $\mathcal{M}_i := \langle S_i, A_i, T_i, R_i, C_i, h \rangle$*, where* S_i*,* A_i*,* $T_i : S_i \times A_i \times S_i \to [0, 1]$*,* $R_i : S_i \times A_i \to \mathbb{R}$*, and* h *are defined as above. Agent i also has a cost function* $C_i : S_i \times A_i \to \mathbb{N}^+$*, which describes how much resource is consumed after an action.*

Cost functions affect the optimisation goal of the agent during policy synthesis. Costs can be considered as *instantaneous constraints* or *budget constraints*:

- Instantaneous constraints are time dependent, e.g., an agent could be constrained such that they can never use more than 3 resources at timestep 0 [de Nijs et al., 2021].
- Budget constraints are considered over the entire time horizon, i.e., they concern the total number of resources used in the agent's execution [de Nijs et al., 2021].

[1] In Puterman [2014], Definition 7 coupled with an expected cost constraint is referred to as a Constrained Markov Decision Process. For clarity, we refer to the model alone as a Markov Decision process with Constraint and the model coupled with the expected cost constraint as a Constrained Markov Decision Process.

For the remainder of this thesis, we will consider budget constraints. We chose budget-constrained MMDP as a first step towards more general cost constraints. Because budget constraints are applied to the full time horizon, they present only one constraint over the optimisation, i.e., a multi-unit resource, whereas instantaneous constraints add a single constraint for each timestep.

For notational convenience we define the random variable $\mathcal{C}_{i,\pi_i}$ to describe the cumulative cost agent i receives when executing policy π_i over the entire time horizon h. With this notation, budget constraints concern the random variable $\mathcal{C}_{i,\pi_i}$. How the $\mathcal{C}_{i,\pi_i}$ should affect the agent's optimisation function is a design choice that is explored in Section 3.1.

2.1.3 Multi-Agent Markov Decision Processes

Multi-Agent Markov Decision Processes (MMDP) are a model to represent sequential decision making in multi-agent systems [Boutilier, 1996].

Definition 8 (Multi-Agent Markov Decision Process [Boutilier, 1996])**.** *A MMDP with n agents is represented by* $\mathcal{M} := \langle S, A, T, R, h \rangle$ *and has joint state space* S*, a factored action space* $A = A_1 \times \cdots \times A_n$*, a joint transition function* $T : S \times A \times S \to [0, 1]$*, a joint reward function* $R : S \times A \to \mathbb{R}$*, and time horizon* $h \in \mathbb{N}^+$ *.*

In MMDPs, agents share a state space, but their actions are individual. The effect of their actions, on the other hand, are dependent on both the shared state space and the actions of other agents. An MMDP can be treated as an MDP, and all MDP solution methods can be applied to MMDPs [Boutilier, 1996] but most solution methods scale exponentially in the number of agents.. Policies are dependent on the joint state space: $\pi : S \to A$ with $\pi(s) = (a_1, a_2, ..., a_n)$. As in the single-agent case, we define the random variable $\mathcal{R}_\pi$ to describe the cumulative reward all agents incurs when executing policy $\pi = \{\pi_i\}_{i \in [n]}$ over the entire time horizon h.

Definition 9 (Factored State Multi-Agent Markov Decision Process [Boutilier, 1996])**.** *Given n independent agents with individual MDPs, we can construct a corresponding MMDP. Suppose each agent* $i \in [n] = \{1, \ldots, n\}$ *has their own finite-horizon*

single-agent MDP $\mathcal{M}_i := \langle S_i, A_i, T_i, R_i, h \rangle$*, and agents share a global time horizon* h*. Then we can construct an MMDP over all* n *agents, represented by* $\mathcal{M} := \langle S, A, T, R, h \rangle$*.* $\mathcal{M}$ *has joint factored state space* $S = S_1 \times \cdots \times S_n$ *and joint factored action space* $A = A_1 \times \cdots \times A_n$*. Let* $s = (s_1, \ldots, s_n)$*,* $a = (a_1, \ldots, a_n)$ *and* $s' = (s'_1, \ldots, s'_n)$*. Then, the joint transition function* $T : S \times A \times S \to [0, 1]$ *is defined by* $T(s, a, s') = \prod_i T_i(s_i, a_i, s'_i)$ *and the joint reward function* $R : S \times A \to \mathbb{R}$ *is defined by* $R(s, a) = \sum_i R_i(s_i, a_i)$*.*

Because there is no coupling between the agents in MMDPs by construction, it is often useful to conduct policy synthesis in the single-agent space and design joint polices that can be factored into single-agent policies: $\pi = \{\pi_i\}_{i \in [n]}$ for a set of single-agent polices π_i.

2.1.4 Multi-Agent Markov Decision Processes with Constraints

We can also extend MMDPs to include constraints.

Definition 10 (Multi-Agent Markov Decision Processes with Constraint [de Nijs et al., 2021])**.** *A* n*-agent (strongly-coupled) MMDP with Constraint is represented by* $\mathcal{M} := \langle S, A, T, R, C, h \rangle$ *and has joint state space* S*, a factored action space* $A = A_1 \times \cdots \times A_n$*, a joint transition function* $T : S \times A \times S \to [0, 1]$*,a joint reward function* $R : S \times A \to \mathbb{R}$*, a joint cost function* $C : S \times A \to \mathbb{R}$*, and a time horizon* $h \in \mathbb{N}^+$ *.*

A MMDP with constraint can be treated as an MDP, and all MDP with constraint solution methods can be applied to MMDPs with constraints as in [Boutilier, 1996].

Definition 11 (Weakly-Coupled MMDP [Meuleau et al., 1998])**.** *Given* n *independent agents with individual MDPs, we can construct the joint weakly-coupled MMDP with constraints. Suppose each agent* $i \in [n] = \{1, \ldots, n\}$ *has their own finite-horizon single-agent MDP* $\mathcal{M}_i := \langle S_i, A_i, T_i, R_i, h \rangle$*, the joint weakly-coupled MMDP with constraints is then represented by* $\mathcal{M} := \langle S, A, T, R, C, h \rangle$ *and has joint state space* $S = S_1 \times \cdots \times S_n$ *and joint action space* $A = A_1 \times \cdots \times A_n$*. Let* $s = (s_1, \ldots, s_n)$*,* $a = (a_1, \ldots, a_n)$ *and* $s' = (s'_1, \ldots, s'_n)$*. Then, the joint transition*

function $T : S \times A \times S \to [0, 1]$ *is defined by* $T(s, a, s') = \prod_i T_i(s_i, a_i, s'_i)$*, the joint reward function* $R : S \times A \to \mathbb{R}$ *is defined by* $R(s, a) = \sum_i R_i(s_i, a_i)$*, and the joint cost function* $C : S \times A \to \mathbb{N}^+$ *is defined by* $C(s, a) = \sum_i C_i(s_i, a_i)$*.*

This definition is a slight variation on the weakly coupled MMDP from Meuleau et al. [1998], which instead uses integral costs and a vector of reward functions. This construction is a *weakly-coupled MMDP* because the only coupling between the individual agents is the shared resource [Meuleau et al., 1998]. In strongly-coupled systems, agents share states, and thus have more direct control over the outcomes of the other agents through action choices. In weakly-coupled systems, they can only affect the other agents by using more or less resources. A MMDP with constraint can be treated as an MDP with constraint or an MMDP with constraint, and all MDP with constraint and MMDP with constraint solution methods can be applied to weakly-coupled MMDPs as in Boutilier [1996]

We focus on weakly-coupled MMDPs throughout this thesis. Weak coupling allows for decentralised policy execution, and often allows for partly decentralised planning. Because strongly-coupled MMDPs have an exponential explosion in the state and action space, planning over weakly-coupled single-agent models multiple times is often significantly faster than planning over the joint, strongly coupled model once, e.g. [Yost and Washburn, 2000].

As with reward, we define the random variable $\mathcal{C}_\pi$ to describe the cumulative cost all agents incurs when executing policy $\pi = \{\pi_i\}_{i\in[n]}$ over the entire time horizon h. Note that in the case of a weakly-coupled MMDP, $\mathcal{C}_\pi = \sum_{i\in[n]} \mathcal{C}_{i,\pi_i}$. As in the single-agent case, there are a variety of ways to consider $\mathcal{C}_\pi$ during policy synthesis. See Section 3.1 for details.

2.1.5 Conditional Value-at-Risk

We now introduce CVaR, a specific measure of uncertainty and risk. CVaR is a useful metric for analysing a distribution, and has many applications in decision making under uncertainty. In this chapter we will use CVaR to constrain the cost distribution

$\mathcal{C}_\pi$. CVaR originally emerged as a way of analysing the tails of a given probability distribution in finance [Rockafellar and Uryasev, 2000, Artzner et al., 1999].

Conditional Value-at-Risk is defined with respect to a concept called Value-at-Risk (VaR):

Definition 12 (Value-at-Risk [Rockafellar and Uryasev, 2000])**.** *Given a random variable Z with cumulative distribution function $F(z)$, the VaR of Z for a given confidence level $\delta \in (0, 1]$ is defined as:*

$$VaR_\delta(Z) = \min\{z | F(z) > 1 - \delta\}. \tag{2.1}$$

VaR can be interpreted as the value at which the $1 - \delta$ quantile of a distribution begins. Now, we can define CVaR.

Definition 13 (Conditional Value-at-Risk [Rockafellar and Uryasev, 2000])**.** *Given a random variable Z and a confidence level $\delta \in (0, 1]$, the CVaR of Z is defined by:*

$$CVaR_\delta(Z) = \frac{1}{\delta} \int_{1-\delta}^{1} VaR_{1-\gamma}(Z) d\gamma. \tag{2.2}$$

Conditional Value-at-Risk is also known as expected shortfall, because it can also be considered as the expected performance in the worst cases, e.g., those cases where the random variable exceeds the VaR. Thus, CVaR can be equivalently defined as:

$$\text{CVaR}_\delta(Z) = E[Z | Z \geq \text{VaR}_\delta(Z)] \quad \text{[Sarykalin et al., 2008].} \tag{2.3}$$

We now focus our attention on a specific type of random variable, $Z = \sum_{j=1}^{n} Z_j$, where each Z_j is an independent random variable. In this context, it can be useful to define the *risk contribution* (RC_i) of each random variable Z_i to $\text{CVaR}_\delta(Z)$ [Tasche, 1999, Yamai and Yoshiba, 2002]. Risk contributions help to decompose risk and assign blame to individual agents.

Definition 14 (Risk Contribution for CVaR [Yamai and Yoshiba, 2002])**.** *The risk contribution of variable Z_i with respect to $Z = \sum_{j=1}^{n} Z_j$ is defined by as:*

$$RC_{\delta,Z}(Z_i) = E[Z_i | Z \geq VaR_\delta(Z)]. \tag{2.4}$$

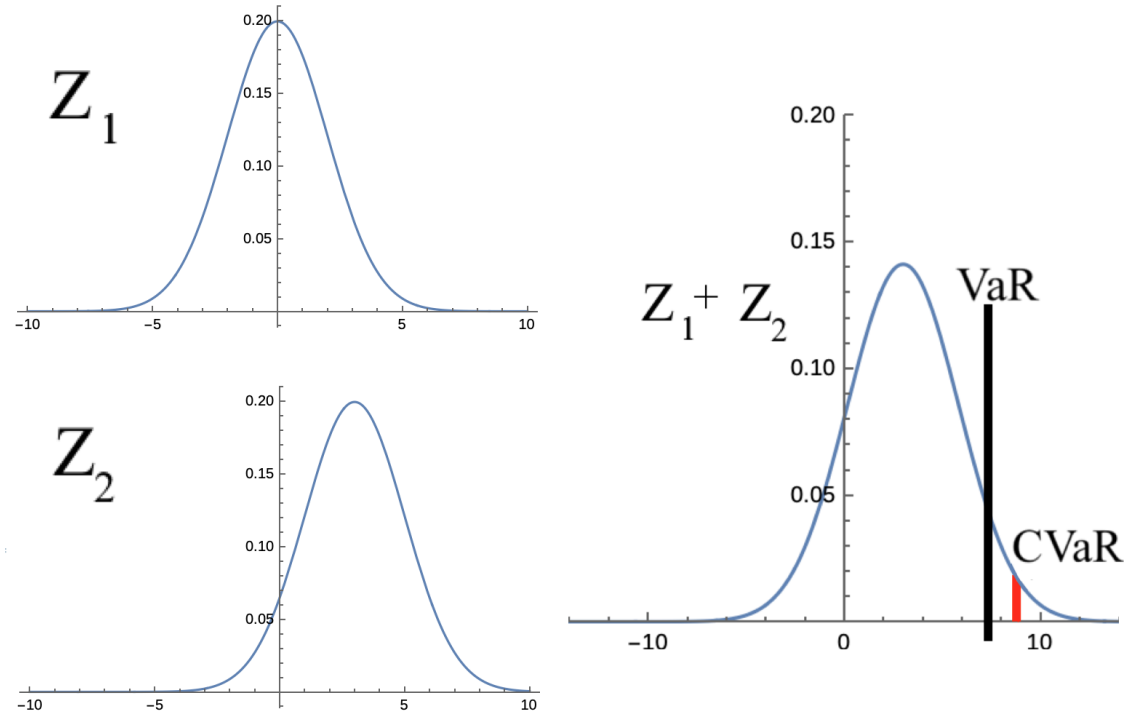


Figure 2.1: Two random variables, $Z_1 \sim \mathcal{N}(\mu = 0, \sigma^2 = 2)$ and $Z_2 \sim \mathcal{N}(\mu = 3, \sigma^2 = 2)$, along with their sum $Z \sim Z_1 + Z_2$. Included is $\text{VaR}_{0.05}(Z)$ and $\text{CVaR}_{0.05}(Z)$.

Intuitively, Z_i's risk contribution is the expected value of variable Z_i in the cases where the joint variable Z exceeds the VaR of Z. It measures how much of the CVaR (i.e., the expected value of random variable Z in the cases where the joint variable Z exceeds the VaR of Z) is due to variable Z_i. As such, risk contribution is defined so that it decomposes the joint CVaR, i.e.:

$$\text{CVaR}_\delta(Z) = \sum_{i=1}^{n} RC_{\delta,Z}(Z_i). \tag{2.5}$$

Example 2. *In Figure 2.1, we illustrate these concepts with the PDFs of distributions* $Z_1 \sim \mathcal{N}(\mu = 0, \sigma^2 = 2)$, $Z_2 \sim \mathcal{N}(\mu = 3, \sigma^2 = 2)$, *and* $Z \sim Z_1 + Z_2$ *with* $\delta = .05$. $VaR = 7.7$ *corresponds to the minimum value of* Z *within the* δ*-tail of* Z. $CVaR = E[Z|Z \geq VaR_\delta(Z)] = 8.8$ *corresponds to the average value of* Z *within the* δ*-tail of* Z. *The risk contribution of variable* Z_1 *is described by:*

$$RC_{\delta,Z}(Z_1) = E[Z_1|Z \geq VaR_\delta(Z)] = 2.9,$$

and the risk contribution of variable Z_2 is described by:

$$RC_{\delta,Z}(Z_2) = E[Z_2|Z \geq VaR_\delta(Z)] = 5.9.$$

We can see that variable Z_2 contributes more risk, which we would expect given its higher mean and equivalent standard deviation.

The random variables under consideration in this chapter are $\mathcal{C}_{i,\pi_i}$ for $i \in [n]$, i.e., the cost distribution that results from agent i executing policy π_i described in Section 2.1.4. Recall that $\mathcal{C}_\pi = \sum \mathcal{C}_{i,\pi_i}$ represents the joint cost distribution over all agents. Additionally, recall that in weakly-coupled MMDPs, the agents' state and action spaces are independent and thus the set of random variables defined by $\{\mathcal{C}_{i,\pi_i} |$ for $i \in [n]\}$ is also independent. Then, $\mathrm{RC}_{\delta,\mathcal{C}_\pi}(\mathcal{C}_{i,\pi_i})$ can be interpreted as the risk that agent i contributes to the system.

2.2 Mechanism Design

Mechanism design is subset of game theory that designs the *rules of encounter* for agents to interact [Myerson, 1989, Hurwicz and Reiter, 2006]. The goal of mechanism design is to create the rules of systems of interaction in such a way that agents acting rationally make decisions that achieve the goals of the system designer. We focus on one subset of mechanism design called auction theory, where an auction designer needs to set up a mechanism to allocate goods from buyers to sellers [Krishna, 2009]. Agents submit bids to the auctioneer which express both what goods an agent is asking for and how much they value those goods. An agent i's value for any subset of goods is known as their *value function* v_i. Each agent submits a variety of bids that express possible options of goods they could be allocated.

For the remainder of the thesis, we will refer to the goods as resources, to match our resource allocation domain. We further narrow our scope to auctions where a single seller has a set of resources to distribute to a set of multiple buyers. Designing an auction in this setting requires a) designing a rule to distribute the goods amongst the buyers and b) determining a set of prices the buyers need to pay for those

goods. In *cooperative* settings, i.e., where all buyers and sellers have the same goal, prices are unnecessary. In *non-cooperative* settings, i.e., where buyers and sellers have differing goals, prices perform an important method for incentivisation. We note that prices can have another function, namely to generate revenue, but that is outside of the scope of this thesis. In the mechanism design community, all auctions are considered in the non-cooperative setting. In spite of this, there is large body of work in multi-agent and multi-robot planning that uses auctions for cooperative decision making, which we will discuss further Section 3.2. Because of this, we separately consider cooperative auctions, as are common in the planning community, and non-cooperative auctions, as are common in the mechanism design community.

2.2.1 Cooperative Auctions

In *cooperative auctions*, the sole problem of the auction designer is construct a rule to allocate the resources among the buyers. This is often referred to as *the winner determination problem* [Lehmann et al., 2006]. It is therefore important to understand the optimisation metric of the auctioneer.

We will focus on social welfare, as it corresponds to the common optimisation problem of reward maximisation described in Section 2.1.1.

Definition 15 (Social Welfare (a.k.a. efficiency, utilitarian welfare) [Krishna, 2009])**.** *Given an auction with n agents, a set of outcomes Ω, and agent valuation functions $v_i : \Omega \to \mathbb{R}^+$ for each $i \in [n]$, the social welfare of an outcome $\omega \in \Omega$ is defined by:*

$$\sum_{i \in [n]} v_i(\omega)$$

Definition 16 (Welfare Maximising Outcome [Krishna, 2009])**.** *Given an auction with n agents, a set of outcomes Ω and agent valuation functions $v_i : \Omega \to \mathbb{R}^+$ for each $i \in [n]$, an outcome $\omega \in \Omega$ is welfare maximising if:*

$$\sum_{i \in [n]} v_i(\omega) \geq \sum_{i \in [n]} v_i(\omega') \quad \forall \omega' \in \Omega$$

Now we focus on two type of auctions with differing resources, *multi-unit auctions* and *combinatorial auctions* .

Multi-Unit Auctions.

Definition 17 (Cooperative Multi-Unit Auctions [Krishna, 2009])**.** *Consider* $L \in \mathbb{N}^+$ *identical resources and* $n \in \mathbb{N}^+$ *agents. Each agent* i *has a value function* $v_i : [L] \to \mathbb{R}^+$ *where* $v_i(k)$ *represents how much agent* i *values* k *resources. A multi-unit auction is a set of rules that determines how to distribute* L *resources amongst the agents.*

Each agent submits a series of *bids* of the form $B_{i,k} = (k, b_{i,k})$, where k is the number of resources they are requesting and $b_{i,k}$ is the bid amount. In the cooperative setting, agents are *truthful* meaning they tell the auctioneer their true value for the resource as their bid, i.e., $b_{i,k} = v_i(k)$ where $v_i(k)$ is the value agent i gives to k resources.

The auctioneer then uses these bids to determine an allocation defined by $\mathcal{F}^\star = \{(k_1^\star, b_1^\star), \ldots, (k_n^\star, b_n^\star)\}$. The resulting allocation is *feasible* if $\sum_{i \in [n]} k_i^\star < L$. When the auctioneer is optimising for social welfare, choosing a feasible, optimal allocation can be expressed as a Integer Linear Program (ILP):

$$\text{maximise} \quad \sum_{i \in [n]} \sum_{k \in [L]} b_{i,k} x_{i,k} \tag{2.6}$$

$$\text{subject to} \quad \sum_{i \in [n]} \sum_{k \in [L]} k x_{i,k} \leq L \tag{2.7}$$

$$\sum_{k \in [L]} x_{i,k} \leq 1 \quad \forall i \in [n] \tag{2.8}$$

$$x_{i,k} \in \{0, 1\} \quad \forall k \in [L], i \in [n] \tag{2.9}$$

Decision variables $x_{i,k}$ correspond to agent i being allocated k resources and are integral, i.e., 0 or 1. The auctioneer directly optimises for the sum of bids (2.6) while constraining the total number of resources (2.7) and ensuring that all agents are allocated a single bid (2.8, 2.9).

Combinatorial Auctions.

Definition 18 (Cooperative Combinatorial Auctions [Cramton et al., 2006])**.** *Consider a set of resources* $\mathcal{Z} = \{z_1, z_2, \ldots, z_m\}$ *and* n *agents. Each subset of items* $A \in 2^{\mathcal{Z}}$ *is known as a* bundle*. Each agent* i *has a well-defined value for every bundle*

$A \in 2^{\mathcal{Z}}$, determined by a value function $v_i : 2^{\mathcal{Z}} \to \mathbb{R}$. A combinatorial auction is a set of rules that determines how to distribute the set of $\mathcal{Z}$ resources amongst the agents.

The agents report a *bid* $B_{i,A} = (A, b_{i,A})$ to the auctioneer for every subset $A \in 2^{\mathcal{Z}}$. In cooperative auctions, agents can be assumed to be truthful, so $b_{i,A} = v_i(A)$. The auctioneer then uses these bids to determine an allocation defined by $\mathcal{F} = \{A_j\}_{j\in[n]}$, where each $A_j \in 2^{\mathcal{Z}}$ represents the bundles allocated to agent j. In the remainder of the thesis, we will use $\{A_j\}$ as shorthand for $\{A_j\}_{j\in[n]}$.

A *feasible allocation* $\{A_j\}$ in this context us one where each agent i is allocated a bundle of items $A_i \in 2^{\mathcal{Z}}$ such that $A_i \cap A_j = \emptyset$ for all agents $j \neq i$ [Cramton et al., 2006]. We denote the set of all feasible allocations by Γ.

When the auctioneer is optimising for social welfare, choosing a feasible, optimal allocation can be expressed as an ILP [De Vries and Vohra, 2003]:

$$\text{maximise} \quad \sum_{i\in[n]} \sum_{A\in 2^{\mathcal{Z}}} b_{i,A} x_{i,A} \tag{2.10}$$

$$\text{subject to} \quad \sum_{\{A\in 2^{\mathcal{Z}} | z\in Z\}} \sum_{i\in[n]} x_{i,A} \leq 1 \quad \forall z \in \mathcal{Z} \tag{2.11}$$

$$\sum_{i\in[n]} x_{i,A} \leq 1 \quad \forall i \in [n] \tag{2.12}$$

$$x_{i,A} \in \{0,1\} \quad \forall A \in 2^{\mathcal{Z}}, i \in [n] \tag{2.13}$$

Decision variables $x_{i,A}$ correspond to agent i being allocated bundle A and are integral. As in the multi-unit auction, the auctioneer directly optimises for the sum of bids (2.10), while ensuring that all agents are allocated a single bid (2.12, 2.13). Then, (2.11) ensures that no item is allocated to more than one agent. For each item $z \in \mathcal{Z}$, there is a constraint that considers all possible bundles that contain z (noted by $\{A \in 2^{\mathcal{Z}} | z \in Z\}$) and ensures that a maximum of one of those bundles is allocated to a single agent.

The winner determination problem for social welfare is NP-complete [Dobzinski and Nisan, 2007] in the number of agents and items.

ILPs are only one way to find the social welfare optimising auction. Another method is the iterative algorithm called iBundle [Parkes, 1999]. iBundle starts with

agents submitting a small subset of their possible bids, in particular those that are the most desirable, i.e, have the highest possible valuation. When a bid enters the auction, the price for that bid is initialised to 0. Then iBundle seeks to find an allocation from the bids that have been proposed with an ILP. If an allocation is found, the auction terminates. If an allocation is not found, the price on every bid is increased by a fixed amount. Then, agents add more bids to the auction. In particular, if agents are acting with a *myopic best response strategy*, they add bids that are now the same or higher valuation as the existing bids given the initial valuation minus the prices of the already added bids. This is repeated until the auction terminates in a valid allocation. iBundle is guaranteed to return the same allocation as the ILP, given agents bid with the myopic best response strategy.

2.2.2 Non-Cooperative Auctions

In non-cooperative auctions, the goals of the auctioneer and individual buyers are not aligned. While the auctioneer wants to optimise for a global goal like social welfare, the goal of the *selfish* individual agent is to maximise their own utility. In the cooperative auction, we assumed all agents were truthful but when agents are selfish, they may lie about their value when presenting bids to the auctioneer.

Example 3. *Consider an auction with one resource, z, and two agents, 1 and 2. Suppose $v_1(z) = 1$ and $v_2(z) = 2$. If agents are truthful, then $b_{1,z} = v_1(z) = 1$ and $b_{2,z} = v_2(z) = 2$. Then the social-welfare-maximising solution is that agent 2 is allocated z, and agent 1 is allocated nothing. But if agent 1 is selfish, they may lie and tell the auctioneer and present $b_{1,z} = 3$ in order to change the end allocation.*

To formally model this type of lying, we require a few definitions.

In the non-cooperative case, each agent has a *strategy function.*

Definition 19 (Strategy Function [Krishna, 2009])**.** *A strategy function $\sigma_i : \mathbb{R} \to \mathbb{R}$ maps from an agent's true value of an item to a bid value to submit to the auctioneer, i.e, agent's bidding function is defined by the composite function $\sigma_i \circ v_i$.*

In the multi-unit case, each agent's bid is uniquely defined by $b_{i,k} := \sigma_i(v_i(k))$ for all $k \in [L]$. In the combinatorial case, each agent's bid is uniquely defined by $b_{i,A} := \sigma_i(v_i(A))$ for all $A \in 2^{\mathcal{Z}}$.

Definition 20 (Dominant Strategy [Krishna, 2009]). *A strategy $\hat{\sigma}_i$ is a dominant strategy for a mechanism if, regardless of what the other agents' bid, agent i receives an equal or higher utility by using strategy $\hat{\sigma}_i$ over any other possible strategy σ_i.*

Definition 21 (Dominant Strategy Equilibrium [Krishna, 2009]). *A dominant strategy equilibrium occurs when all agents play a dominant strategy.*

A fundamental aspect of non-cooperative auctions is that agents are rational (strategic).

Proposition 1. *A rational, or strategic, agent chooses a dominant strategy, if one exists.*

With this context, rational agent lies when their dominant strategy is something other than the identity function. As we see in Example 3, lying can change the end allocation of the mechanism so that it results in a sub-optimal allocation. This is a problem from the perspective of the auctioneer, whose goal is optimality. To address this problem, the auctioneer can introduce a set prices $p_1, ..., p_n$ that agents need to pay. Well designed prices can modify agents' incentives so they obtain the highest utility when the are truthful. The price an agent pays is a design feature of the mechanism, and is dependent on the final allocation $\{A_j\}$. If prices depend only on the allocation, they are called *anonymous*, if they depend on both the agent and the allocation, they are called *discriminatory*.

When prices are introduced, agents' utility functions are defined not just by how much they value the item, but also how much they have to pay. Formally, agent i's *utility* function for their allocation is defined by $u_i(A_i) := v_i(A_i) - p_i$.

Example 4. *Returning to the scenario from Example 3, suppose all agents' prices are set to 0, i.e., $p_i = 0\ \forall i$. Consider agent 1's dominant strategy $\hat{\sigma}_1$, which needs to be constructed so that it is the best strategy no matter what agent 2 does. Because*

agent 2 could bid an arbitrarily high number, $\hat{\sigma}_1 := \infty$. The same is true for agent 2. In this case, the auctioneer will receive two bids $b_{1,z} = \infty$ and $b_{2,z} = \infty$. Both bids are independent of the true agent valuations of the resource, and thus they give the auctioneer no useful information to use toward determining the social-welfare-maximising allocation. The best the auctioneer can do in this case is to randomly allocate the resource.

Prices need to be constructed so they modify the agents' utilities in such a way that the agents are incentivised to give useful information to the auctioneer. This property is known as *incentive compatibility.*

Definition 22 (Incentive Compatibility [Krishna, 2009])**.** *A mechanism is incentive compatible if telling the truth is a dominant strategy. More formally, this means that all agents' dominant strategies $\hat{\sigma}_i$ are defined by the identity function. Then, $\sigma_i(v_i) = v_i$.*

Incentive compatible mechanisms are useful because if agents truthfully report their valuation as their bids, the auctioneer can effectively optimise for a global goal. Because, by definition, $b_{i,A} := v_i(A)$ in incentive compatible auctions, the auctioneer can assume that each agent's valuation for a set of items is equivalent to their bid. Then, the problem of maximising for a global goal is the same as in the cooperative setting, where agents also reported $b_{i,A} := v_i(A)$.

There is one more important property in non-cooperative auctions, called *individual rationality*, which incentivises agents to participate.

Definition 23 (Individual Rationality [Krishna, 2009])**.** *A mechanism is individually rational if each agent is better off participating in the auction than they would be otherwise, meaning their utility after allocation is non-negative.*

We now introduce a multi-unit and combinatorial auction that is both incentive compatible and individually rational.

2.2.3 The Vickrey-Clarke-Groves Auctions

One auction that satisfies the property of incentive compatibility is the Vickrey-Clarke-Groves (VCG) auction, named after ideas combined from [Vickrey, 1961, Clarke, 1971, Groves, 1973]. . In VCG, the auctioneer allocates bundles based on the *welfare-maximising* outcome described in Section 2.2.1.

In VCG, prices p_i are set as follows, as in Cramton et al. [2006]. Let $\mathcal{F}$ be the welfare-maximising outcome, and let V be the combined valuation of the agents that are not i in $\mathcal{F}$. Let $\mathcal{F}^{-i}$ be the welfare-maximising outcome if agent i was not involved in the auction, i.e., agent i is allocated the empty set, and let V^{-i} be the combined valuation of the agents that are not i under $\mathcal{F}^{-i}$. Then, $p_i = V - V^{-i}$. $\mathcal{F}^{-i}$ represents a counterfactual of what would occurred if agent i did not participate in the system. And thus the payment p_i then represents the amount that agent i has perturbed the system, as agent i pays the difference between the total welfare of all other agents if they had not been in the auction minus the welfare of all other agents when they are included in the auction. Now, we explicitly state the VCG prices applied to both multi-unit and combinatorial auctions.

Definition 24 (Multi-Unit VCG Prices [Cramton et al., 2006].)**.** *Consider a multi-unit auction with a final welfare-maximising allocation* $\mathcal{F}^\star = \{(k_1^\star, b_1^\star), \ldots, (k_n^\star, b_n^\star)\}$. *Let* $\mathcal{F}^{-i} = \{(k_1^{-i}, b_1^{-i}), \ldots, (k_n^{-i}, b_n^{-i})\}$ *be the welfare-maximising auction if agent i was not included in the auction. Then, the multi-unit VCG price* p_i *is defined by*

$$p_i = \sum_{j \in [n] \setminus \{i\}} b_j^{-i} \; - \sum_{j \in [n] \setminus \{i\}} b_j^\star.$$

Definition 25 (Combinatorial VCG Prices [Cramton et al., 2006])**.** *Consider a combinatorial auction with a final welfare-maximising allocation* $\mathcal{F}^\star = \{A_j^\star\}$. *Let* $\mathcal{F}^{-i} = \{A_j^{-1}\}$ *be the welfare-maximising allocation if agent i was not included in the auction. Then, the combinatorial VCG price* p_i *is defined by*

$$p_i = \sum_{j \in [n] \setminus \{i\}} b_{j, A_j^{-i}} \; - \sum_{j \in [n] \setminus \{i\}} b_{j, A_j^\star}.$$

Proposition 2. *VCG is both incentive compatible and individually rational. [Cramton et al., 2006]*

In combinatorial auctions, only optimal solvers can be provide incentive compatible solutions [Nisan and Ronen, 2007]. Heuristic methods cannot be incentive compatible, though can be difficult to manipulate.

2.3 Discussion

2.3.1 Limitations of our Modelling Choices

Throughout the thesis we make two important modelling choices: restricting to pre-allocated policies and restricting to finite-horizon problems. We address the reasons for these two choices below.

We consider the problem of pre-allocation for a few reasons. First, pre-allocation allows for decentralized execution. Agents need to communicate during planning time, but during execution, they only need knowledge of their own state space and pre-allocated policy to make action choices. Second, pre-allocation allows agents to plan within the confines of a set amount of resources, and optimally act for themselves given the allocated amount of resources. When planning online for constrained resources, agents' actions are coupled by the online planning, which might result in agents having to make sacrifices during execution due to the realised uncertainty or failure of the other agents. If agents had known in advanced they were going to need to make these sacrifices, they might have made different initial action choices. As a result, we consider pre-allocation to be more fair for independent agents. There are downsides to this approach, namely that agents cannot correct for failures. In fully autonomous systems, this presents a problem, but in practice, many autonomous systems, like driver assist cars and factory robots, have human operators to handle failure cases.

Finite-horizon problems allow us to easily plan for a fixed, finite number of a given resource. While there are a few ways to consider constrained resources over infinite horizon, these approaches come with their own problems, detailed here. First, one can discount the cost, as is common with reward. This means that resources used in the future 'count for less' than resources used now. This is mathematically useful, but has important implications, especially when pre-allocating resources. If

agents never re-plan, they will be able to use arbitrarily high resources at some point in the future (though how far in the future is proportional to the discount factor). If the resource constraints are important, this is unsuitable. Another option is to consider the long-run average cost (similar to a long run average reward) of an infinite horizon policy. For finite MDPs, memoryless policies will eventually reach steady-states, and we can instead consider the expected cost accumulated during the steady-states. This construction is useful in tasks like monitoring, where agents need to repeatedly achieve the same goal, but it is less useful in tasks where agents' goals change over time. It also suffers from the opposite problem of discounted cost, in that the initial trace before reaching the steady state disappears in the limit, and thus is ignored in the optimisation. As a result, it is possible to construct an MDP that uses an arbitrarily high number of resources before reaching the steady state. So, like discounted cost, this is unsuitable when resource constraints are important. Finally, one could generate infinite horizon policies for a non-discounted cumulative cost. This type of policy is not guaranteed to exist, and would consist of finding a finite horizon policy that adheres to the cost constraint, and a steady state policy that consumes no resources. We leave consideration of this option to future work.

2.3.2 Summary

In this Chapter, we summarised the requisite background in planning under uncertainty and mechanism design. In the next Chapter, we will discuss the relevant literature in the same categories, along with the baselines and domains which will compare our work to in Chapters 4 and 5.

3
Literature Review

3.1 Cooperative Resource Allocation Under Uncertainty

In this section we summarise the literature on preplanning over single and multi-agent MDPs with Constraints. In particular, we focus on *budget-constrained* resources, i.e., resources that are constrained in summation over a time horizon [de Nijs et al., 2021].

First, we focus on single-agent methods. There are a variety of ways an agent can choose to constrain their resource usage. Consider that every fixed offline policy has a corresponding distribution over possible costs that the agent will incur when executing that policy, which we defined by $\mathcal{C}_{i,\pi_i}$ in Section 2.1. This occurs as a result of the uncertainty within the system; pre-set actions may result in different outcomes due to the stochasticity of the environment, and these differing outcomes may require different future resource usage. The resulting distribution over possible costs can be handled in different ways when an agent tries to constrain their resource usage.

Worst-Case Constraint In one setting, the entire distribution must be bounded by some resource limit L. In other words, a policy satisfying a worst-case constraint must always use less than L resources, no matter how the uncertainty is resolved. In terms of $\mathcal{C}_{i,\pi_i}$, we can describe this constraint as:

$P_{\pi_i}[\mathcal{C}_{i,\pi_i} \leq L] = 1$. This is a restrictive constraint, and in some settings may be impossible to achieve.

Expected-Case Constraint Another option is an expected constraint, where the agent considers the expectation of the cost distribution and bounds that value by some resource limit L, as in [Altman, 1999]. Altman [1999] introduce the Constrained Markov Decision Process (CMDP) which ensures that the expected cost is contained by some L. In terms of $\mathcal{C}_{i,\pi_i}$, we can describe this constraint as: $E_{\pi_i}[\mathcal{C}_{i,\pi_i}] \leq L$. Altman [1999] also introduce a Linear Program (LP) based solution method which develops an occupation measure for each state-time pair. This allows the agent more flexibility when planning, but provides no formal guarantees or information on the distribution of the cost accumulated. Because it disregards the variability of resource usage between possible outcomes, there may be a high probability that the resource limit is violated on any given run.

Chance Constraint Another common approach is planning with a chance-constraint [Haskell and Jain, 2015, Santana et al., 2016, Ayton and Williams, 2018], which limits the percent of cases which exceed some resource consumption limit L. In terms of $\mathcal{C}_{i,\pi_i}$, we can describe this constraint as: $P_{\pi_i}[\mathcal{C}_{i,\pi_i} > L] \leq \delta$. Chance-Constrained MDPs (CCMDPs) allow for constraint violations in unlikely scenarios, but still guarantee that the constraint is met with a specified likelihood. In other words, a chance-constraint ensures that everything *except* the δ-tail of the cost distribution is bounded by L. Haskell and Jain [2015] introduce an LP solution method for policy generation in CCMDPs that utilises a convex analytic method. Santana et al. [2016] design a heuristic search algorithm for chance-constrained Partially Observable MDPs (POMDPs). Giuseppi and Pietrabissa [2020] and L. A. and Fu [2022] propose reinforcement learning approaches for CCMDPs. Ayton and Williams [2018] introduce a Monte Carlo tree search based algorithm for large CCMDPs. Recent applications of single-agent CCMDPs include: autonomous vehicles [Huang

et al., 2018], motion primitive planning under parametric uncertainty [Gutow and Rogers, 2021], and spacecraft decision making [Timmons et al., 2021]. However this approach provides no understanding over how bad the worst-cases get because it ignores the distribution within the δ-tail.

Risk Constraint Finally, the constraint on resource consumption can be formulated to bound the *risk* within the cost distribution. Risk-constraints reason over how bad the worst-cases of the distribution are, and one such example of a risk constraint is Conditional Value-At-Risk (CVaR) in Borkar and Jain [2014], which solves a Risk-Constrained MDP (RCMDP). When considering CVaR, the agent constrains the expected cost within the δ-tail of the distribution and bounds that value by some resource consumption limit L. In terms of $\mathcal{C}_{i,\pi_i}$, we can describe this constraint as: $\text{CVaR}(\mathcal{C}_{i,\pi_i}) \leq L$. This method allows for a formal analysis of the worst-cases of the cost distribution.

MMDPs with constraints extend these concepts to a multi-agent system. In this case, the distribution over resource usage is described by the possible outcomes of the sum of the agents' cost functions under a given joint which we defined by $\mathcal{C}_\pi$. Any of the methods described above can be directly applied to MMDP with constraints by considering the strongly coupled joint model as in Boutilier [1996], but this approach scales poorly, as the state and action spaces of an MMDP are exponential in the number of agents. Hence, much research has focused on reasoning over individual agent models separately and considering only the global constraint jointly. This is often referred to as a **weakly-coupled** MMDP with constraints [Meuleau et al., 1998] as discussed in Section 2.1.4. In this case $\mathcal{C}_\pi = \sum_{i\in[n]} \mathcal{C}_{i,\pi_i}$. Again, the global constraint can be considered in a number of ways.

Worst-Case Constraint Agrawal et al. [2016] present an approximate worst-case solution, which uses a decentralised greedy algorithm to allocate resources. Wu and Durfee [2010] present a MILP for planning in situations where resources are strictly constrained by ensuring that the budget is not exceeded even in the worst case.

Expected-Case Constraint Column Generation solves the weakly-coupled Constrained MMDP (CMMDP) problem in a decentralised manner through an iterative LP based algorithm that solves a series of non-constrained single-agent MDPs [Walraven and Spaan, 2018, Yost and Washburn, 2000].

Chance Constraint de Nijs et al. [2017] extend the chance-constrained problem to MMDPs, which we will refer to as Chance-Constrained MMDPs (CCMMDPs). They present an algorithm which uses Hoeffding bounds to artificially lower the resource limit to the point that solving for the new limit with traditional expected-case methods also ensures that the original chance constraint is met. This approach works best in settings where the number of agents is very large, since it allows for the lowered limit to approach the original limit as the Hoeffding bound gets tighter. de Nijs et al. [2017] is also applicable to instantaneous constraints.

For an in-depth overview of multi-agent resource allocation techniques, including techniques designed for instantaneous resources and online resource allocation, see de Nijs et al. [2021].

3.1.1 Conditional Value-at-Risk

There is a wealth of literature that studies the risk associated with sequential decision making under uncertainty in both single-agent and multi-agent settings, and in particular the risk measure CVaR. We classify it into four categories: approaches which (1) minimise the CVaR of a cost function and contain no constraint, (2) minimise the expectation of a cost function and constrain the CVaR of that same function, (3) maximise the CVaR of a reward function and constrain the CVaR of a cost function, and (4) maximise the expectation of a reward function and constrain the CVaR of a cost function. Note that (4) matches the domain of MDP and MMDPs with constraints that was described in the previous section. Here, we consider both classical planning and reinforcement based methods.

(1) Most of the current literature is focused on category (1), minimising CVaR directly, with no additional constraints. This category is well studied in both classical planning [Chow et al., 2015, Yu et al., 2018, Li et al., 2021b] and reinforcement learning [Tamar et al., 2015a,b, Keramati et al., 2020, Qiu et al., 2021, Rigter et al., 2021]. These approaches differ from our approach (and the previous MDP with constraints literature) in that the primary goal is to minimising some cost, instead of optimising for reward while minimising a cost. This is ill suited to resource allocation domains, where resources are a constraint, not an optimisation function.

(2) Category (2), minimising the expectation of a cost function while also constraining the CVaR of that same cost function, has been studied in classical planning [Chow and Ghavamzadeh, 2014, Rigter et al., 2022] and reinforcement learning [Hiraoka et al., 2019, Ying et al., 2022]. These methods cannot be adapted to use different functions in the optimisation and constraint, e.g., maximising for a reward function while constraining the CVaR of a cost function as we require. In the resource allocation domain, it is important to have both a reward function that models the agents' goals and a constrained cost function that models agents' resource usage.

(3) In category (3), maximising the CVaR of a reward function and constraining the CVaR of a cost function, Ahmadi et al. [2021] does consider both a reward function and a cost function, but considers the CVaR of both the reward and cost constraint, which allows for similar solution methods to category (2).

(4) Category (4) maximises the expectation of a reward function and constrains the CVaR of a cost function; we call this the risk-constrained MDP (RCMDP) planning problem. Borkar and Jain [2014] devised an algorithm to solve the RCMDP planning problem for a single-agent. The methodology of this approach is discussed in Section 3.3.1. Chow et al. [2017] and L. A. and Fu [2022] solve a similar problem with single-agent reinforcement learning.

3.2 Auctioning in Multi-Agent Planning

Auctioning approaches have been widely used in robotics, particularly for cooperative coordination of teams of robots, but also, to a lesser extent, for non-cooperative planning.

Early work in cooperative auctions focus on allocating tasks among teams of robots. Hunsberger and Grosz [2000] used combinatorial auctions to distribute tasks across a set of agents in cooperative scenarios. Gerkey and Mataric [2002] used first-price one-round auctions in the same context. Later, in Lagoudakis et al. [2005], a sequential single item (SSI) auctioning mechanism was proposed for multi-robot routing and task allocation. SSIs combine the advantages of parallel single-item auctions and combinatorial auctions, achieving good quality task allocation with low computational effort [Koenig et al., 2006]. SSI auctioning has also been extended in Nunes and Gini [2015] to handle temporal constraints, in McIntire et al. [2016] to handle precedence constraints, and in Street [2022] to handle spatiotemporal uncertainty. From the perspective of the agents participating in SSI auctions, Tovey et al. [2005] investigated the problem of generating different bids taking into account different global objectives to be achieved. Capitan et al. [2013] distribute tasks under uncertainty by considering role policies for POMDPs, whilst Schillinger et al. [2018] considers MDPs, auctioning subtasks such that a global Linear Temporal Logic (LTL) task is achieved by the team. Los et al. [2020] consider a decentralised combinatorial auction for the problem vehicle routing in cooperative systems. As we discuss in Chapter 6, problems such as vehicle routing can be considered as a resource allocation problem.

In our work the robots are *competing* for the use of these resources rather than *cooperating* to achieve a set of tasks. In the context of planning under uncertainty, Dolgov and Durfee [2006] use a Generalised Vickrey Auction to allocate resources in an MMDP with a worst-case constraint. Bererton et al. [2003] present an auction based approach for the management of shared resources with MMDPs with an expected-case constraint. De Nijs et al. [2015] considers an auction based arbitrage method for allocating resources, but the arbitrage occurs online during

execution, and thus requires constant communication. In many environments, synchronous communication between all agents cannot be assumed, like in outdoor environments, under water, or in jurisdictions communications might be regulated, or even intercepted [Gielis et al., 2022].

Amir et al. [2015] and Chandra et al. [2022] consider non-cooperative auctions in the context of multi-agent pathfinding. Brafman et al. [2009] introduce a game-theoretic model that can be adopted to path planning via STRIPS, but it assumes that agents are willing to form coalitions.

Finally, non-cooperative multi-agent planning exists outside of auctions and within the context of stochastic games. Stochastic games are a model for non-cooperative multi-agent interaction on non-constrained MMDPs. Stochastic games are strongly coupled, meaning the actions of any one agent has direct impact on the reward functions of the other agents. The goal is to find *equilibrium* strategies of players. Stochastic games exist in both reinforcement learning [Littman, 1994] and planning [Kearns et al., 2000, Niu and Clark, 2019]. Solan and Vieille [2015] provide a comprehensive overview of applications for stochastic games. Mechanism design on two-stage stochastic games, introduced in Ieong et al. [2007], is a promising area of research. More recent work designs an incentive compatible mechanism [Satchidanandan and Dahleh, 2022], and applies mechanism design on two-stage stochastic games to optimise reward while constraining CVaR in energy markets [Li et al., 2017]. As of yet, applicability is limited to two-stage stochastic games, i.e., MMDPs with a time horizon of 2.

3.3 Discussion

We now summarise the relevant baselines and domains that will be used throughout the thesis.

3.3.1 Planning Under Uncertainty

Multi-Agent Planning Baselines

We consider four prominent baselines for MMDPs with Constraints.

The **Constrained Multi-Agent Markov Decision Process** (CMMDP) method from Altman [1999] solves the CMMDP problem, which optimises for expected reward while constraining the expected cost:

Definition 26 (Constrained Multi-Agent MDP Altman [1999])**.** *A Constrained Multi-Agent MDP (CMMDP) is a triple* $\langle \mathcal{M}, L \rangle$*.* $\mathcal{M}$ *denotes a weakly-coupled MMDP.* $L \in \mathbb{N}^+$ *denotes the limit of the globally constrained resource (i.e., total resource use of all agents over all timestep is limited by* L*)*

The goal of the CMMDP is to find a joint policy $\pi^\star = \{\pi_i^\star\}_{i \in [n]}$ that maximises the cumulative reward, subject to limiting the expected cost:

$$\pi^\star = \underset{\pi := \{\pi_i\}_{i \in [n]}}{\arg\max} \quad E_\pi \left[\sum_{i \in [n]} \mathcal{R}_{i,\pi_i} \right], \tag{3.1}$$

$$\text{s.t.} \quad E_\pi \left[\sum_{i \in [n]} \mathcal{C}_{i,\pi_i} \right] < L. \tag{3.2}$$

This centralised approach considers individual agents' MDPs expanded to include the current timestep. The solution method consists of an LP which optimises for expected reward. Variables in the LP are occupancy measures, which reflect the probability that, given the synthesised policy, agent i is in state s at time t taking action a. The first set of constraints ensures the transition dynamics of the MMDP are met. Then, the last constraint corresponds to ensuring that the resource limit is met in expectation.

The **Model-Checker Multi-Agent Markov Decision Processes** (MMDP) method consists of solving for the chance-constrained fully-coupled MMDP.

Definition 27 (A (fully-coupled) Constrained Multi-Agent MDP [de Nijs et al., 2021])**.** *A fully-coupled Chance-Constrained Multi-Agent MDP (CCMMDP) is a triple* $\langle \mathcal{M}, L, \delta \rangle$*.* $\mathcal{M}$ *denotes a fully-coupled MMDP.* $\delta \in [0, 1)$ *denotes the limit on the probability of constraint violation.* $L \in \mathbb{N}^+$ *denotes the limit of the globally*

constrained resource (i.e., total resource use of all agents over all timesteps is limited by L) – a resource constraint violation occurs when this is limit is exceeded, i.e.,

$$\sum_{i\in[n]}\sum_{t\in[h]} C_i(s_{i,t}, a_{i,t}) > L. \tag{3.3}$$

This type of constraint is sometimes referred to as a budget constraint [de Nijs et al., 2021]. Then the goal of the *fully-coupled* CCMMDP is to find a joint policy $\pi^\star$ that maximises the cumulative reward, subject to the chance constraint:

$$\pi^\star = \arg\max_{\pi} \quad E_\pi\left[\sum_{i\in[n]} \mathcal{R}_{i,\pi}\right], \tag{3.4}$$

$$\text{s.t.} \quad P_\pi\left[\sum_{i\in[n]} \mathcal{C}_{i,\pi} > L\right] < \delta. \tag{3.5}$$

To solve the chance-constrained problem, we extend the state space of the fully-coupled MMDP described in the previous chapter to include cumulative cost. Then we generate a Pareto frontier with a probabilistic model checker over expected reward and the probability of exceeding the resource limit L. We then choose the two points with probability of exceeding closest to δ and mix the corresponding deterministic policies to generate a stochastic policy which exactly meets the chance constraint. This provides an optimal solution for the chance-constrained problem, but due to the size of the joint state space it is often infeasible.

The chance-constrained **Column Generation** (CG) method solves a (weakly-coupled) Chance-Constrained Multi-Agent MDP (CCMMDP)

Definition 28 (A (weakly-coupled) Constrained Multi-Agent MDP [de Nijs et al., 2021])**.** *A (weakly-coupled) Chance-Constrained Multi-Agent MDP (CCMMDP) is defined by a triple $\langle \mathcal{M}, L, \delta\rangle$. $\mathcal{M}$ denotes a weakly-coupled MMDP. $\delta \in [0, 1)$ denotes the limit on the probability of constraint violation. $L \in \mathbb{N}^+$ denotes the limit of the globally constrained resource (i.e., total resource use of all agents over all timesteps is limited by L) – a resource constraint violation is defined as above.*

The weakly-coupled CCMMDP problem was introduced in [de Nijs et al., 2017]. Then the goal of the weakly-coupled CCMMDP is to find a joint policy $\pi^\star = \{\pi_i^\star\}_{i\in[n]}$

that maximises the cumulative reward, subject to the chance constraint:

$$\pi^\star = \underset{\pi:=\{\pi_i\}_{i\in[n]}}{\arg\max} \quad E_\pi \left[\sum_{i\in[n]} \mathcal{R}_{i,\pi_i} \right], \tag{3.6}$$

$$\text{s.t.} \quad P_\pi \left[\sum_{i\in[n]} \mathcal{C}_{i,\pi_i} > L \right] < \delta. \tag{3.7}$$

The CG methodology from [de Nijs et al., 2017] combines two techniques. First, it uses column generation which solves the same problem as the CMMDP but in a decentralised manner, as in Yost and Washburn [2000]. Like the CMMDP method, it uses an LP that optimises for expected reward and ensures that the resource limit is met in expectation. This method results in stochastic policies made up of a mixture of deterministic policies for each agent, with the same optimal reward, total cost, and probability of exceeding L as the CMMDP method. Because it is decentralised, it is much faster than CMMDP. The second technique that the chance-constrained CG uses is a Hoeffding bound, as in de Nijs et al. [2017]. L_{hoeff} is an artificially lowered resource limit that is based on L, δ and the maximum resource usage of each agent. In the LP, L_{hoeff} is used instead of L, so the algorithm returns policies limited by L_{hoeff} in expectation. L_{hoeff} is designed so this will also limit the chance constraint, i.e., if the policies uses less than L_{hoeff} resources in expectation, then those same policies will exceed L with probability less than δ. The CG method solves the CCMDP problem, albeit conservatively.

Because L_{hoeff} can be too conservative, de Nijs et al. [2017] suggest a method to dynamically relax L_{hoeff}. After each run of CG, the **Dynamic Column Generation** (DCG) method runs Monte Carlo trials to estimate the probability distribution over resources and modify L_{hoeff} accordingly. This process is repeated until the chance constraint is met without slack.

Risk-Constrained Planning Baselines

We consider one prominent single-agent risk-constrained baseline.

iRMDP is an algorithm introduced in Borkar and Jain [2014] which introduces and solves the single-agent risk-constrained planning problem.

Definition 29 (Risk-Constrained MDP)**.** *A Risk-Constrained MDP (RCMDP) is a triple* $\langle \mathcal{M}_i, L, \delta \rangle$*.* $\mathcal{M}_i$ *denotes the single-agent MDP.* $\delta \in (0, 1]$ *denotes the confidence level.* $L \in \mathbb{N}^+$ *denotes the limit of the globally constrained budget resource, which is to be constrained by CVaR:*

$$CVaR_\delta \left[\mathcal{C}_{i,\pi_i}\right] < L. \tag{3.8}$$

The goal of the risk-constrained MDP is to find a policy $\pi^\star$ that maximises the cumulative reward, subject to the risk constraint:

$$\pi_i^\star = \arg\max_{\pi_i} \quad E_{\pi_i}\left[\mathcal{R}_{i,\pi_i}\right], \tag{3.9}$$

$$\text{s.t.} \quad \text{CVaR}_\delta \left[\mathcal{C}_{i,\pi_i}\right] < L. \tag{3.10}$$

Because our single-agent policy update algorithm in Section 5.2.4 modifies iRMDP, we include for the reader a extended summary of the method. Note that, unlike Borkar and Jain [2014], we describe iRMDP below specifically for discrete state-space MDPs to match the literature on MMDPs with Constraints, though the readers should note the iRMDP algorithm does generalise to continuous MDPs. The iRMDP algorithm solves the single-agent risk-constrained planning problem using a Lagrangian relaxation:

$$\min_{\lambda \geq 0} \max_{\pi_i} E_{\pi_i}\left[\mathcal{R}_{i,\pi_i}\right] + \lambda \left[L - \text{CVaR}_\delta \left[\mathcal{C}_{i,\pi_i}\right]\right]. \tag{3.11}$$

The algorithm to optimise for Equation 3.11 is described in Algorithm 1. The procedure has an outer loop that iteratively updates λ until the constraint is satisfied (lines 2-18).

For a given λ^w, the following procedure is used to calculate the optimal value function $J_t(s, y)$. Here, y corresponds to the cost that has been accumulated so far. Thus, $J_t(s, y)$ is defined as:

$$J_t(s, y) = \max_{\pi_i} \quad E_{\pi_i}\left[\sum_{\tilde{t}=t}^{h} R_{i,\pi_i}(s, c, \tilde{t})\right] + \lambda^w \left[L - \text{CVaR}_\delta \left[y + \sum_{\tilde{t}=t}^{h} C_{i,\pi_i}(s, c, \tilde{t})\right]\right]. \tag{3.12}$$

Algorithm 1 iRMDP [Borkar and Jain, 2014]

Require: A single-agent MDP $\mathcal{M}_i := \langle S_i, h, A_i, T_i, R_i, C_i \rangle$, an initial state s_0, a risk constraint L, and a confidence bound $\delta \in (0, 1]$

1: $\lambda^0 = 0$
2: **for** w=1,2,...,until converged **do**
3: $\quad \beta^{w,0} = 0$
4: $\quad$ **for** v=1,2,...,until converged **do**
5: $\quad\quad J_{h+1}[s, y] = \lambda^w (L - \frac{y}{\delta} \mathbb{1}_{(y > \beta^{w,v})})$
6: $\quad\quad V_{h+1}[s, y] = \mathbb{1}_{(y > \beta^{w,v})}$
7: $\quad\quad Q_{h+1}[s, y] = \frac{y}{\delta} \mathbb{1}_{(y > \beta^{w,v})}$
8: $\quad\quad$ **for** $t = h, h-1, ..., 0$ **do**
9: $\quad\quad\quad G_t^{w,v}[s, y, a] = R_i(s, a) + \sum_{s'} T_i(s, a, s') J_{t+1}^{w,v}[s', yC_i(s, a)]$
10: $\quad\quad\quad J_t^{w,v}[s, y] = \max_{a \in A} G_t^{w,v}[s, y, a]$
11: $\quad\quad\quad \pi_t^{w,v}[s, y] = \arg\max_{a \in A} G_t^{w,v}[s, y, a]$
12: $\quad\quad\quad V_t^{w,v}[s, y] = \sum_{s'} T_i(s, \pi_t^{w,v}[s, y], s') V_{t+1}^{w,v}[s', y + C_i(s, \pi_t^{w,v}[s, y])]$
13: $\quad\quad\quad Q_t^{w,v}[s, y] = \frac{C_i(s, \pi_t^{w,v}[s,y]) V_t^{w,v}[s,y]}{\delta} + \sum_{s'} T_i(s, \pi_t^{w,v}[s, y], s') Q_{t+1}^{w,v}[s', y + C_i(s, \pi_t^{w,v}[s, y])]$
14: $\quad\quad$ **end for**
15: $\quad\quad \beta^{w,v+1} = \beta^{w,v} - \frac{1}{v}(\delta - V_0^{w,v}[s_0, 0])$
16: $\quad$ **end for**
17: $\quad \lambda^{w+1} = (\lambda^w - \frac{1}{w}(L - Q_0^{w,v}[s_0, 0]))^+$
18: **end for**

$J_t(s, y)$ is the sum of the future payoffs received by Equation 3.11 when executing the optimal policy between time t and h, starting at state s and having already accumulated y cost. Then, $J_0(s_0, 0)$ describes the value of the optimal policy. Note that the policies returned by iRMDP depend on both time and cumulative cost.

Because the optimal value function $J_t(s, y)$ contains the CVaR of a policy, it cannot be calculated directly via value iteration. CVaR corresponds to the expected value in the δ-tail, which is dependent on VaR. CVaR cannot be calculated without knowing VaR, and the two cannot be calculated concurrently in a single-iteration of value iteration. So, in order to correctly solve for the terms which include CVaR, VaR must be calculated first in the middle loop (lines 3-16). VaR is guessed with an initial $\beta^{w,0} = 0$, which is iteratively updated until it accurately reflects the VaR of the synthesised policy.

Finally, the inner loop (lines 8-14) conducts value iteration to synthesis a policy π_i along with policy evaluation on $P_{\pi_i}[\mathcal{C}_{i,\pi_i} \geq \beta^{w,v}]$ and $E_{\pi_i}[\mathcal{C}_{i,\pi_i} | \mathcal{C}_{i,\pi_i} \geq \beta^{w,v}]$. Line

9-10 calculates $J_t(s, y)$ with value iteration for a fixed VaR defined by $\beta^{w,v}$, and line 11 extracts a time dependent policy $\pi_{i,t}$ from $J_t(s, y)$, Line 12 uses policy evaluation to calculate $V_t(s, y)$ where $V_t(s, y)$ is the probability that the cumulative resource between timestep (t, h), plus the already accumulated y cost, is at least $\beta^{w,v}$ when executing π_i starting in state s. Then, $V_0(s_0, 0) = P_{\pi_i}[\mathcal{C}_{i,\pi_i} \geq \beta^{w,v}]$. Line 13 uses policy evaluation to calculate $Q_t(s, y)$ where $Q_t(s, y)$ is the expected value of the cumulative resource between timestep (t, h), plus the already accumulated y cost, when the that same value is at least $\beta^{w,v}$ when executing π_i starting in state s. Then $Q_0(s_0, 0) = E_{\pi_i}[\mathcal{C}_{i,\pi_i} | \mathcal{C}_{i,\pi} \geq \beta^{w,v}]$.

Line 15 then iteratively updates $\beta^{w,v}$ until it converges to to the VaR, which occurs when $\delta = P_{\pi_i}[\mathcal{C}_{i,\pi_i} \geq \beta^{w,v}] = V_{t=0}(s, y = 0)$. Once this occurs, $Q_0(s_0, 0)$ becomes equal to the CVaR of the current policy π_i. Then, the line 17 iteratively updates λ^w until convergence ensures that risk constraint is met exactly.

Domains

We consider three important domains from the constrained planning under uncertainty literature.

First, we consider the **Maze Domain** from Wu and Durfee [2010]. Agents operate in a grid world that represents the surface of Mars, with 40% of grid cells chosen at random to represent untraversable terrain, and 10% of cells chosen at random to represent places at which reward can be obtained by completing a task. Tasks further away from the start position result in a higher reward. An example of the Maze domain on a 8 by 8 grid is shown in Figure 3.1. We first modified the Maze domain to use a multi-unit resource. Agents have two types of actions: regular actions, which consume no resource, but only move to their intended location 40% of the time; and safe actions, which consume one resource and move to their intended locations 95% of the time.

An example of a safe action is a movement action coupled with a localisation subroutine, which greatly improves a robot's chances of moving in the correct direction, but also consumes additional battery power. Once an agent is in a task

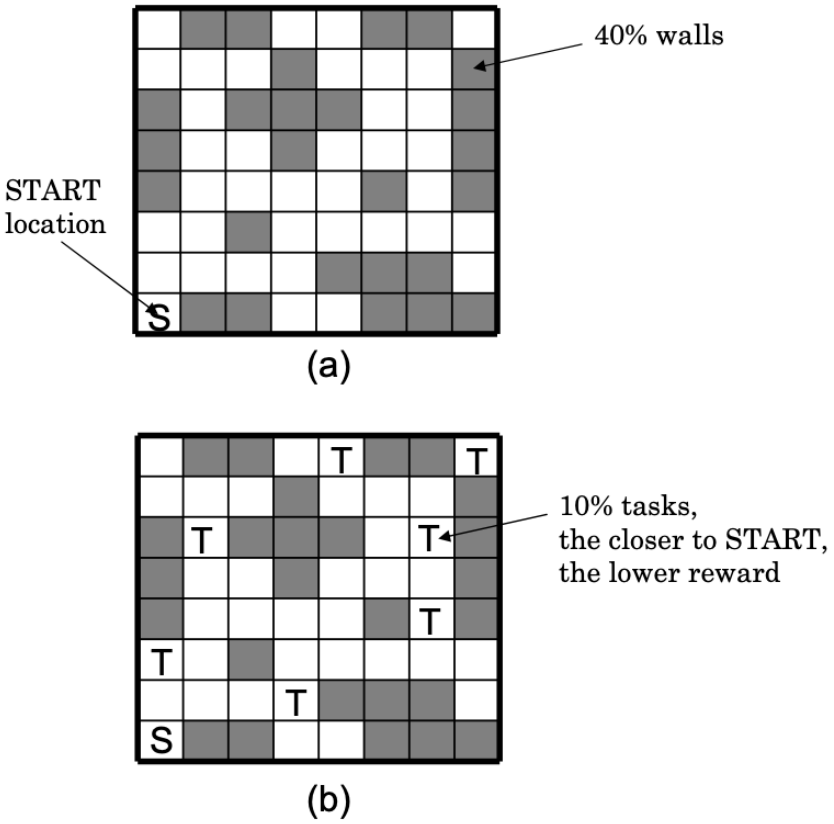
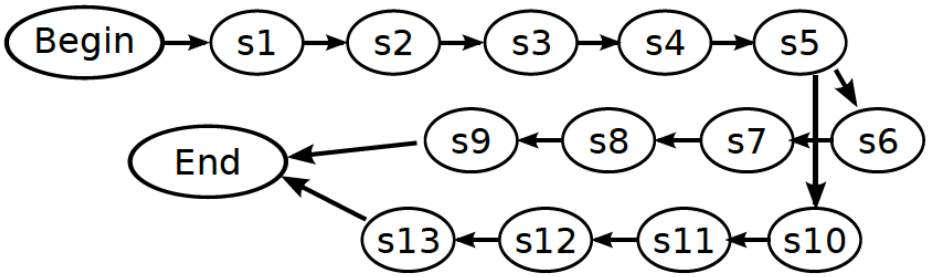


Figure 3.1: An example of the Maze domain on a 8 by 8 grid, taken from Wu and Durfee [2010]. "The procedure of creating a random grid world. (a) 40% of the locations are randomly chosen as walls. (b) 10% of the locations are randomly chosen for tasks."

location, they can choose to perform a task action, at which point their execution ends. Agents' only interaction with each other comes in the form of a global resource constraint, which is set to L.

Next, we consider the **Synthetic Advertising Domain** from Boutilier and Lu [2016], where an advertiser must choose a strategy to allocate a monetary budget among 1000 different agents to convert documented interest into sales. The MDP is illustrated in Figure 3.2. Each agent is described by the 15 state MDP which can be split into 3 parts: generic interest in the product category (States 1-5); interest in the advertiser's specific product (States 6-9): or, interest in a competitor's product (States 10-13). The advertiser has five advertising actions at each state, which



Search: s1: unint; s2: general int; s3: search1, s4: search2, s5:search3
Advertiser: s6: interest1; s7: interest2; s8: interest3, s9: conversion
Compt'r: s10: interest1; s11: interest2; s12: interest3, s13: conversion

Figure 3.2: Synthetic Advertising Domain MDP from Boutilier and Lu [2016].

start at a zero-cost/no intervention actions and increasingly become more costly, and more effective at moving the consumer toward purchasing the advertiser's product. All actions are stochastic, having some probability of the agent exiting the process. In states 1-4, more costly actions have a higher likelihood of moving the consumer to become more interested in the generic product. In State 5, more costly actions have a higher probability of moving toward State 6, where the consumer has a documented interest in the advertiser's product, over State 10, where the consumer has a documented interest in the competitor's product. If the consumer does move to State 6, higher cost actions give them a better chance of moving toward State 9, where they purchase the product. If the consumer moves to State 10 or above, higher cost actions have a better probability of moving the agent *backwards* in the states, away from purchasing the competitor's product and toward State 5, where they can instead move to be interested in the advertiser's product. When a consumer purchases either the competitor's or advertiser's product, they also exit the process. The advertiser is rewarded when an agent purchases their product. All agents' MDPs are identical but independent, so the advertiser can pursue different strategies for different agents.

Finally, we consider the **Autonomous Driving** domain from Rigter et al. [2022]. This domain uses real world traffic data from the Caltrans Performance

Measurement System (PeMS) data set, which deployed over 39,000 real-time traffic sensors throughout California's metropolitan areas. We considered a set of 30 roads, some of which are freeways and some of which are local roads. The time taken to traverse a road is stochastic, with transition probabilities gathered from the data set. As in Rigter et al. [2022], highways also contain random noise generated to simulate low-probability traffic jams. This is a high cost-variability environment, with single action costs ranging from 1 to over 400.

We now define this problem in the context of MMDPs with constraints. Each agent starts and ends at a random location in the graph. We optimise for the number of agents that successfully reach their goal, while limiting the probability that the joint time it takes for the agents to reach their goal exceeds some maximum travel time limit L. This means that agents receive a reward of 100 when they reach their goal. After an agent takes a road, and the time taken to traverse the road is realised, agents receive a cost corresponding to the travel time. Note that the cost is not defined by the number of timesteps in their MDP, but instead by the *travel times* as defined by the PeMs data set. With this optimisation metric, the global resource allocation problem chooses which agents go and complete a task, and the single-agent planning problem is choosing what route to take to minimise the constraint.

4

Allocating Chance-Constrained Resources

4.1 Introduction

We first consider the problem of allocating resources under uncertainty in both cooperative and non-cooperative multi-agent systems. For example, consider a research station on Mars, which comprises multiple robots operated by researchers. The robots might have to share access to electricity to power operation, or network bandwidth to send data back to Earth. In this chapter, we focus on discrete, *multi-unit resources* like battery power and bandwidth. Our goal is to design a system that preallocates the resources among the agents. This is a simple resource type, but the problem becomes more difficult when designing a resource distribution system in this context that considers the fact that agents are working in an uncertain environment. In this chapter, we focus on uncertainty that can be modelled with a Markov Decision Process (MDP). This can include uncertainty that arises from the environment that the robot encounters or from the physical execution of the robot's actions.

Because agents cannot fully control their environment, taking the same actions may result in a different outcome which may, in turn, require different future resource use. This results in agents that are uncertain of the quantity of resources they will consume in a given day in pursuit of their tasks, even if they follow a fixed plan. Because resources are preallocated before the uncertainty over resource

Table 4.1: A description of Chapter 4's relevance to the three main challenges of multi-agent decision making.

	Chapter 4	Chapter 5	Chapter 6
Uncertain Domain?	Yes	Yes	No
Rich Uncertainty?	No	Yes	No
Multiple Resource Types?	No	No	Yes
Non-Cooperative?	Yes	No	Yes

usage is resolved, as we discussed in Section 3.1. the system must decide how to deal with probabilistic information. If the system plans for the worst-case (i.e., assuming that uncertainty is resolved with maximum resource usage), resources may be underutilised in practice. This occurs because agents must avoid any possibility (however unlikely) of using too many resources. On the other hand, planning only for expected resource usage could result in agents counting on resources without any assurance that they are actually available. For example, suppose the robots receive power from a shared solar panel each day; our aim would be to design a system that preallocates the available power among the different robots. It is essential that, in *most* cases, the total power use of the robots does not exceed the amount of energy generated. Say a robot started the day with 100% of their onboard battery power, and was preallocated enough solar power to use and recharge 50% of their onboard battery. While they may plan to only use 50% of their battery during daily tasks, stochasticity in the environment may cause them to use 60% of their battery instead. But this need not result in a system failure – it only means that the robot would need to draw an extra 10% out of the solar power. If no extra power is available, they would start their day with only 90% battery. Never taking this opportunity would be excessively risk averse, but doing so repeatedly would risk flat batteries. Because of this single-agent uncertainty over resource consumption, we consider a global system that plans for *chance constraints*: that is, we seek to limit the probability of resource violations. This problem was originally introduced in de Nijs et al. [2017], and we restate it as a Chance-Constrained Multi-Agent MDP (Section 4.1.1).

In this chapter, we present the **Auction for Chance-Constrained Resources** (ACCR), an auction mechanism which handles uncertainty over resource usage.

In ACCR, agents are asked to bid on resources, reporting both how much they value each resource and how likely they are to exceed a given resource bound. An Integer Linear Program (ILP) *allocates resources* to agents while ensuring a chance constraint over the probability of exceeding the resource limit is met over all agents (Section 4.2). We then consider how multi-objective reasoning can be used to *generate single-agent bids* (Section 4.3). We also *present a pricing structure* that can be used to apply the mechanism to non-cooperative settings (Section 4.4). Finally, we *empirically evaluate ACCR* against state-of-the-art approaches using Maze [Wu and Durfee, 2010], a classic multi-agent constrained resource domain, an advertising budget allocation domain [Boutilier and Lu, 2016], and a real world driving domain [Rigter et al., 2022]. Our results in outperform the state of the art for up to hundreds of agents (Section 4.6).

This chapter describes the work published in Gautier et al. [2023a] in the Proceedings of AAAI 2023.

4.1.1 Chance-Constrained Multi-Agent MDPs.

We first re-reintroduce CCMDP problem from Section 3.3.1.

Definition 30 (Chance-Constrained Multi-Agent MDP)**.** *A (weakly-coupled)Chance-Constrained Multi-Agent MDP (CCMMDP) is a triple* $\langle \mathcal{M}, L, \delta \rangle$*.* $\mathcal{M}$ *denotes a weakly-coupled MMDP.* $\delta \in [0, 1)$ *denotes the limit on the probability of constraint violation.* $L \in \mathbb{N}^+$ *denotes the limit of the globally constrained resource (i.e., total resource use of all agents over all timesteps is limited by L) – a resource constraint violation occurs when this is limit is exceeded, i.e.,*

$$\sum_{i \in [n]} \sum_{t \in [h]} C_i(s_{i,t}, a_{i,t}) > L. \tag{4.1}$$

This type of constraint is sometimes referred to as a budget constraint [de Nijs et al., 2021] and the CCMMDP problem was introduced in [de Nijs et al., 2017]. Then the goal of the CCMMDP is to find a joint policy $\pi^\star = \{\pi_i^\star\}_{i \in [n]}$ that maximises

the cumulative reward, subject to the chance constraint:

$$\pi^\star = \underset{\pi:=\{\pi_i\}_{i\in[n]}}{\arg\max} \quad E_\pi \left[\sum_{i\in[n]} \mathcal{R}_{i,\pi_i} \right], \tag{4.2}$$

$$\text{s.t.} \quad P_\pi \left[\sum_{i\in[n]} \mathcal{C}_{i,\pi_i} > L \right] < \delta. \tag{4.3}$$

where $\mathcal{R}_{i,\pi_i}$ to describe the cumulative reward agent i receives when executing policy π_i over the entire time horizon h, and $\mathcal{C}_{i,\pi_i}$ to describe the cumulative cost agent i receives when executing policy π_i over the entire time horizon h, as defined in Section 2.1.

4.2 Auction for Chance-Constrained Resources

We now present the Auction for Chance-Constrained Resources (ACCR), which takes into account individual agent's uncertainty in resource consumption to better allocate chance-constrained resources. To do this, we modify the multi-unit auction described in Section 2.2.1. In a multi-agent scenario, it is important for agents to understand how many resources they can use before choosing and executing a policy. With this in mind, the mechanism is executed as follows. At the beginning of a cycle, agents calculate bids that correspond to policies and submit them to the auctioneer. Next, the auctioneer carries out the auction protocol described below to allocate the resources. Then, the agents proceed with policies that correspond to their allocated resource amount. Single-agent planning and execution are decentralised, but the auctioneer acts a centralised arbitrator to decide which agents get what resources.

4.2.1 The ACCR Protocol.

In ACCR, each agent submits a list of tuples of the form $B_{i,\alpha} = (k_{i,\alpha}, b_{i,\alpha}, \epsilon_{i,\alpha})$. As in the traditional multi-unit auction, $k_{i,\alpha}$ denotes the number of resources and $b_{i,\alpha}$ represents the amount that agent i would be willing to pay for $k_{i,\alpha}$ resources. Unlike the traditional multi-unit auction, agents also submit $\epsilon_{i,\alpha}$, which corresponds to the probability that, if agent i is allocated $k_{i,\alpha}$ resources, they will exceed this resource limit during execution. Agents can submit multiple bids for the

same number of resources, as long all bids have a different values for b and ϵ. Implicit to each bid $B_{i,\alpha}$ is a policy $\pi_{i,\alpha}$ which agent i would execute if they were allocated $k_{i,\alpha}$ resources. Thus, $b_{i,\alpha}$ represents the expected reward of policy $\pi_{i,\alpha}$ (i.e., $E_{\pi_{i,\alpha}}[R_{i,\pi_{i,\alpha}}]$) and $\epsilon_{i,\alpha}$ represents the probability that the total number of resources consumed while executing $\pi_{i,\alpha}$ exceeds $k_{i,\alpha}$ (i.e., $P_{\pi_{i,\alpha}}[C_{i,\pi_{i,\alpha}} > k]$). While a policy $\pi_{i,\alpha}$ is privately identified with each bid $B_{i,\alpha}$, no policies (or MDPs) are revealed to the auctioneer. Agent bid generation is described in the next section, and each agent i submits m_i bids.

The auctioneer solves the winner determination problem with the following Integer Nonlinear Program (INLP).

$$\text{maximise} \quad \sum_{i\in[n]} \sum_{\alpha\in[m_i]} b_{i,\alpha} x_{i,\alpha} \tag{4.4}$$

$$\text{subject to} \quad \sum_{i\in[n]} \sum_{\alpha\in[m_i]} k_{i,\alpha} x_{i,\alpha} \leq L \tag{4.5}$$

$$\prod_{i\in[n]} \left(1 - \sum_{\alpha\in[m_i]} \epsilon_{i,\alpha} x_{i,\alpha}\right) \geq 1 - \delta \tag{4.6}$$

$$\sum_{\alpha\in[m_i]} x_{i,\alpha} \leq 1 \quad \forall i \in [n] \tag{4.7}$$

$$x_{i,\alpha} \in \{0,1\} \quad \forall \alpha \in [m_i], i \in [n] \tag{4.8}$$

Decision variables $x_{i,\alpha}$ correspond to agent i being allocated $k_{i,\alpha}$ resources. As in a traditional multi-unit auction, the auctioneer directly optimises for the sum of bids (4.4) while constraining the total number of resources (4.5) and ensuring that all agents are allocated a single bid (4.7, 4.8). In ACCR, the auctioneer also has access to agents' declared probabilities of exceeding each bid. To plan for the chance constraint, the auctioneer assumes all agents' probabilities of exceeding are independent. This assumption follows from the definition of weakly-coupled MDPs with Constraints from Section 2.1.4. This is because agents' probabilities of exceeding are dependent only on their own policy, cost function, and transition function, all of which are independent in weakly-coupled MMDPs with Constraints. Satisfying (4.6) is equivalent to ensuring that the chance constraint in (4.3) holds. To see this, note that because the joint declared resource use of all agents is less

than L, equation (4.3) is bounded by the probability that none of the agents exceed their declared resource use. Because agents' resource usages are independent, the probability that none of the agents exceed their declared resource use is equivalent to the product of the probability that each agent does not exceed their declared resource use, which results in (4.6). Constraint (4.6) is nonlinear, which makes the problem an INLP. However, it can be translated into a linear constraint. Towards this goals, we prove the following Lemma, which relies on the fact that for any agent, the LP variable for one bid is exactly 1, and the LP variables for all other bids are 0. This then allows for a transformation of the product to a sum through the properties of logarithms.

Lemma 3. *Under Constraints (4.7) and (4.8), the following holds:*

$$\sum_{i\in[n]}\left(\log\left(1-\sum_{\alpha\in[m_i]}\epsilon_{i,\alpha}x_{i,\alpha}\right)\right)=\sum_{i\in[n]}\sum_{\alpha\in[m_i]}x_{i,\alpha}\log\left(1-\epsilon_{i,\alpha}\right)$$

Proof. First, we want to show that for each $i \in n$,

$$\log\left(1-\sum_{\alpha\in[m_i]}\epsilon_{i,\alpha}x_{i,\alpha}\right)=\sum_{\alpha\in[m_i]}x_{i,\alpha}\log\left(1-\epsilon_{i,\alpha}\right) \tag{4.9}$$

Constraint (4.7) and (4.8) taken together imply that:

- $x_{i,\alpha} \in \{0, 1\}$, for all i, α; and
- at most one element from $\{x_{i,\alpha} | \alpha \in [m_i]\}$ can be equal to 1, for all $i \in n$.

This means that for an arbitrary agent $i \in [n]$, either:

a) $x_{i,\alpha} = 0$ for all $\alpha \in [m_i]$, or,

b) there exists an $\tilde{\alpha} \in [m_i]$ such that $x_{i,\tilde{\alpha}} = 1$ and $x_{i,\alpha} = 0$ for all $\alpha \neq \tilde{\alpha}$.

Case a:

$x_{i,\alpha} = 0$ for all $\alpha \in [m_i]$ implies that:

$$\begin{aligned}\log\left(1-\sum_{\alpha\in[m_i]}\epsilon_{i,\alpha}x_{i,\alpha}\right)&=\log\left(1-\sum_{\alpha\in[m_i]}\epsilon_{i,\alpha}\cdot 0\right)\\&=\log(1)\\&=0,\end{aligned}$$

and,

$$\begin{aligned}\sum_{\alpha\in[m_i]} x_{i,\alpha}\log\left(1-\epsilon_{i,\alpha}\right) &= \sum_{\alpha\in[m_i]} 0\cdot\log\left(1-\epsilon_{i,\alpha}\right)\\ &= 0.\end{aligned}$$

So in Case a, Equation 4.9 is true.

Case b:

Because there exists an $\tilde{\alpha}\in[m_i]$ such that $x_{i,\tilde{\alpha}}=1$ and $x_{i,\alpha}=0$ for all $\alpha\neq\tilde{\alpha}$:

$$\begin{aligned}&\log\left(1-\sum_{\alpha\in[m_i]}\epsilon_{i,\alpha}x_{i,\alpha}\right)\\ &\quad=\log\left(1-\left(\epsilon_{i,\tilde{\alpha}}x_{i,\tilde{\alpha}}+\sum_{\substack{\alpha\in[m_i]\\ \alpha\neq\tilde{\alpha}}}\epsilon_{i,\alpha}x_{i,\alpha}\right)\right)\\ &\quad=\log\left(1-\left(\epsilon_{i,\tilde{\alpha}}\cdot 1+\sum_{\substack{\alpha\in[m_i]\\ \alpha\neq\tilde{\alpha}}}\epsilon_{i,\alpha}\cdot 0\right)\right)\\ &\quad=\log\left(1-\epsilon_{i,\tilde{\alpha}}\right),\end{aligned}$$

and,

$$\begin{aligned}&\sum_{\alpha\in[m_i]} x_{i,\alpha}\log\left(1-\epsilon_{i,\alpha}\right)\\ &\quad= x_{i,\tilde{\alpha}}\log\left(1-\epsilon_{i,\tilde{\alpha}}\right)+\sum_{\substack{\alpha\in[m_i]\\ \alpha\neq\tilde{\alpha}}} x_{i,\alpha}\log\left(1-\epsilon_{i,\alpha}\right)\\ &\quad= 1\cdot\log\left(1-\epsilon_{i,\tilde{\alpha}}\right)+\sum_{\substack{\alpha\in[m_i]\\ \alpha\neq\tilde{\alpha}}} 0\cdot\log\left(1-\epsilon_{i,\alpha}\right)\\ &\quad=\log\left(1-\epsilon_{i,\tilde{\alpha}}\right).\end{aligned}$$

So in Case b, Equation 4.9 is true.

Therefore, we can conclude that Equation 4.9 is true for arbitrary $i\in[n]$, and

thus:

$$\sum_{i\in[n]}\left(\log\left(1-\sum_{\alpha\in[m_i]}\epsilon_{i,\alpha}x_{i,\alpha}\right)\right) = \sum_{i\in[n]}\sum_{\alpha\in[m_i]}x_{i,\alpha}\log\left(1-\epsilon_{i,\alpha}\right)$$

under Constraints (4.7) and (4.8). □

Now we can design an ILP with an equivalent solution.

Proposition 4. *Substituting Constraint (4.6) in the above INLP with the following Constraint (4.10) will result in an ILP with an equivalent solution:*

$$\sum_{i\in[n]}\sum_{\alpha\in[m_i]}x_{i,\alpha}\left(\log\left(1-\epsilon_{i,\alpha}\right)\right)\geq\log\left(1-\delta\right). \tag{4.10}$$

Thus replacing Constraint (4.6) in our original INLP with Constraint (4.10) results in an equivalent ILP that solves the chance-constrained allocation problem more efficiently than the original INLP.

We denote the allocation yielded by solving the ILP as:

$$\mathcal{F}^\star=\{(k_1^\star,b_1^\star,\epsilon_1^\star),\ldots,(k_n^\star,b_n^\star,\epsilon_n^\star)\}.$$

4.3 Single-Agent Decision Making

ACCR requires that each agent can generate bids of the form $B_{i,\alpha}=(k_{i,\alpha},b_{i,\alpha},\epsilon_{i,\alpha})$. To do so, they must compute a list of possible policies π_α, and determine those policies' respective resource uses, values, and probabilities of exceeding $k_{i,\alpha}$ resources. A procedure to do this is detailed below.

Given their respective MDP $\mathcal{M}_i=\langle S_i,A_i,T_i,R_i,C_i,h\rangle$, agents first need to extend their MDP to a new MDP $\tilde{\mathcal{M}}_i:=\langle\tilde{S}_i,A_i,\tilde{T}_i,\tilde{R}_i,\tilde{C}_i,h\rangle$. The new state space is defined by $\tilde{S}_i=S_i\times[h]\times\mathbb{N}^+$, which allows $\tilde{\mathcal{M}}_i$ to include the current timestep and the current cumulative cost in the state space in order to reason over the probability of exceeding a certain resource bound k. The initial state of the MDP, described by $s_{i,0}$, is extended to $\tilde{s}_{i,0}=(s_{i,0},0,0)$, as agents start their execution at time 0 having consumed no resources. Any state $\tilde{s}$ in the extended MDP can be

decomposed into $\tilde{s} = (s, t, c)$ where s is a state in the original MDP, t is a timestep, and c is the current cumulative cost at that timestep. The set of actions remains the same. The reward $\tilde{R}_i : \tilde{S}_i \times A_i \to \mathbb{R}$ is defined as $\tilde{R}_i((s,t,c),a) = R_i(s,a)$ and similarly the cost $\tilde{C}_i : \tilde{S}_i \times A_i \to \mathbb{N}^+$ is defined as $\tilde{C}_i((s,t,c),a) = C_i(s,a)$. The transition function is extended to $\tilde{T}_i : \tilde{S}_i \times A_i \times \tilde{S}_i \to [0,1]$:

$$\tilde{T}_i(\tilde{s},\tilde{a},\tilde{s}') = \begin{cases} T_i(s,a,s') & \text{if} \\ & \tilde{s} = (s,t,c), \text{ and} \\ & \tilde{s}' = (s', t+1, c + C(s,a)) \\ 0 & \text{otherwise.} \end{cases} \tag{4.11}$$

Agent i reasons over their extended MDP to solve a **multi-objective** problem [Roijers et al., 2013]. Such problems are concerned with simultaneously optimising for two or more distinct objectives. In the case of our agents, for any given number of resources $k \leq L$, they are interested in policies that:

1. increase their cumulative reward, and
2. decrease their probability of exceeding k.

We first address how to optimise for the objectives individually.

Increasing their Cumulative Reward: The first objective agents need to plan when considering bids is their primary optimisation objective, optimising for their cumulative reward. The optimisation objective asks: can we find a policy that maximises the cumulative reward? This is expressed as:

$$\max_{\pi_i} E_{\pi_i}\left[\tilde{\mathcal{R}}_{i,\pi_i}\right]. \tag{4.12}$$

Probabilistic Model Checkers are a common method of optimising reward maximisation queries. Suitable model checkers include PRISM [Kwiatkowska et al., 2002], which we use in our evaluation, and STORM [Dehnert et al., 2017]..

Decreasing their Probability of Exceeding: The second objective is a reachability property. The optimisation objective asks: can find a policy that minimises

the probability that we eventually reach a state where the cumulative cost exceeds k? In logic, F is the symbol for eventually. Thus, this is expressed as:

$$\min_{\pi_i} P_{\pi_i} \left[\mathrm{F}\ c > k \right]. \tag{4.13}$$

This problem can be equivalently expressed as maximising a safety property. In this case, the optimisation objective asks: can find a policy that maximises the probability that we never reach a state where the cumulative cost exceeds k? Or equivalently, can find a policy that maximises the probability that the cumulative cost is always less that or equal to k? In logic, G is the symbol for always. Thus, this is expressed as:

$$\max_{\pi_i} P_{\pi_i} \left[\mathrm{G}\ c \leq k \right]. \tag{4.14}$$

We will instead consider the formulation in Objective 4.14, as it is easier to conceptualise the trade-off required by two maximisation properties.

Now we address how to optimise for these objective simultaneously.

Multi-Objective Optimisation: These objectives are conflicting, meaning that improving one objective may come at the expense of the other. One common way to understand optimality in multi-objective problems is with **Pareto optimality**. A policy is Pareto optimal if there exists no other policy that strictly improves one objective without worsening the others. The set of such policies is called the **Pareto frontier**. The Pareto frontier represents the trade-offs that arise for optimising for one objective over the other.

Given a fixed k, we can build a Pareto frontier that optimises for Objectives (4.12) and (4.14).

We propose that agents generate their bids by generating one Pareto frontier for each $k \leq L$. Bidding using this Pareto frontier is reasonable because it represents optimal trade-offs between reward maximisation and the probability of overconsumption of the resource. When optimising for these two objectives, each point along the Pareto frontier corresponds to a policy π_i. The Pareto frontier can be represented by a set of *deterministic* policies [Forejt et al., 2012]. For each such

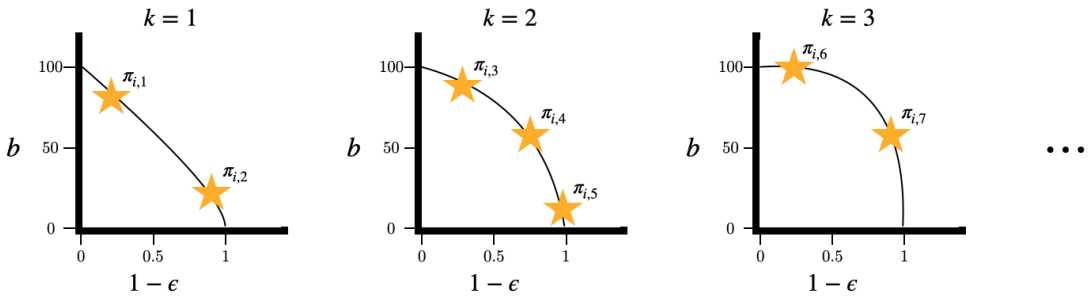


Figure 4.1: An example of the Pareto frontiers generated by our single-agent decision making algorithm. Each curve corresponds to a set number of resources k. The points along the Pareto frontiers, represented by stars, correspond to deterministic polices. The coordinates of each policy instruct the agent what ϵ and b to submit to the auctioneer.

policy, the agent can generate a bid (k, b, ϵ) where k corresponds to the resource use corresponding to the current Pareto frontier, b corresponds to the current Pareto point's value for (4.12), and $1-\epsilon$ corresponds to the current Pareto point's value for (4.14). An example of these Pareto frontiers is shown in Figure 4.1. In this example, stars correspond to deterministic policies. For example, for policy $\pi_{i,7}$, the agent would submit the following bid to the auctioneer: $(k = 3, b = 50, \epsilon = 0)$. Note that as the number of resources increase, the agent is always able to obtain the same or higher bid value for the same $1-\epsilon$ because they now how more resources available. Also note that for any k, when $1-\epsilon$ is 0, i.e., the agent always exceeds their resource, the agent is able to receive their maximum possible reward, which in this case is 100.

After each agent generates a list of bids, they are sent to the auctioneer. We require that agents only send bids with $\epsilon < 1$. This is required so that the auctioneer can compute $\log(1-\epsilon)$ in the ILP. This requirement is without loss of generality because bids with $\epsilon = 1$ would give no useful information to the auction, essentially saying only that the agent is *guaranteed to exceed* the stated number of resources. The number of bids generated depends on the maximum number of resources, since a Pareto frontier is generated for each possible k. It also depends on the number of deterministic policies along the Pareto frontier. The agents can also rule out points with $\epsilon > \delta$ as they would not be accepted by an auctioneer. Agents can lower their

computation time by submitting fewer bids to the auctioneer, either by sending only a subset of the deterministic policies that represent the Pareto frontier, or by generating fewer Pareto frontiers (e.g., only generating frontiers for k divisible by 5).

4.4 Extending to the Non-Cooperative Case

Since our approach extends the multi-unit auction to cases with uncertainty, we can also extend the non-cooperative version of the multi-unit auction to create a non-cooperative ACCR. We define a non-cooperative resource allocation problem as one where the auctioneer's goal is to optimise Equation (4.2) constrained by Equation (4.3), but the goal of each agent differs. Individual agents are self-interested, so the goal of agent i is to choose a policy π_i^* that maximises their own reward:

$$\pi_i^* = \arg\max_{\pi_i} E_{\pi_i}\left[\sum_{t\in[h]} \mathcal{R}_{i,\pi_i}\right] \tag{4.15}$$

Agents can maximise their reward by lying about how much they value resources during bidding, as discussed in Section 2.2.2. To prevent this strategy, non-cooperative auctions use prices to incentivise agents to tell the truth. The non-cooperative multi-unit auction we discussion is Section 2.2 is called a Vickrey-Clarke-Groves (VCG) auction, which can be used to allocate resources to self-interested agents *without* the presence of uncertainty [Dobzinski and Nisan, 2010, Vickrey, 1961, Clarke, 1971, Groves, 1973]. The price structure of VCG is designed so that agents are incentivised to give accurate information to the auctioneer.

We can extend this price structure to ACCR. To calculate prices, the auctioneer first calculates what the prices $p_1^v, p_2^v, \ldots, p_n^v$ would be paid in a traditional multi-unit VCG auction, as described is Section 2.2. The ACCR prices are charged after each agent executes their private policies $\pi_i^\star$ associated to $(k_i^\star, b_i^\star, \epsilon_i^\star)$, as they depend on the realised use of each agent, which we refer to as k_i^r. If agent i is allocated k_i resources by the auctioneer and does not exceed that limit after executing through time horizon h, then they do not pay a price. If they do exceed their allocated number of resources (either because of the inherent uncertainty

in their models, or because they chose to modify their policy to a more resource-heavy one) they are charged a higher price based on the usual VCG price and their reported probability of exceeding:

$$p_i = \begin{cases} p_i^{\perp} & \text{if } k_i^r \leq k_i^\star \\ p_i^{\top} & \text{otherwise.} \end{cases} = \begin{cases} 0 & \text{if } k_i^r \leq k_i^\star \\ \dfrac{1}{\epsilon_i^\star} \cdot p_i^v & \text{otherwise.} \end{cases} \tag{4.16}$$

This price is designed so that the price paid is p_i^v in expectation. This pricing structure ensures that agents are best off if they truthfully report their value b, and are best off if they do not under-report ϵ. To ensure that agents do not over-report ϵ, the auctioneer can run a statistical test to determine if any agent is consistently over-reporting ϵ over a series of runs of the auction. We elaborate on these properties next.

4.4.1 Theoretical Properties of the Price Structure

We motivate why the price structure and secondary statistical analysis incentivise agents to tell the truth to the auctioneer. In each bid, the agents report two pieces of information that could be untruthful, a bid value b and a probability of exceeding the declared number of resources ϵ. First, we show that if ϵ is truthfully reported, then b is truthfully reported. Next we provide a secondary mechanism to ensure ϵ is truthfully reported.

Lemma 5. *Given a strategic agent i and bid $B_{i,\alpha} = (k, b, \epsilon)$ with implied policy π_i, if ϵ is truthfully reported, then $b = v_i(k)$.*

Proof. Suppose that ϵ is reported truthfully, and let k^r be the realised resource use of policy π_i. Then the price that an agent pays in expectation is the same as the traditional VCG payment:

$$\begin{aligned} E_{\pi_i}[p_i] &= P_{\pi_i}[k_r \leq k] \cdot p_i^{\perp} + P_{\pi_i}[k_r > k] \cdot p_i^{\top} \\ &= (1 - \epsilon) \cdot 0 + \epsilon \cdot \left(\frac{1}{\epsilon} \cdot p_i^v \right) \\ &= p_i^v \end{aligned}$$

Because the expected price is equivalent to the price in a traditional VCG auction, which is compatible, each agent's best option is to truthfully report their value $v_i(k)$ as their bid $B_{i,\alpha}$. □

Lemma 6. *Given a strategic agent i and bid $B_{i,\alpha} = (k, b, \tilde{\epsilon})$, for all b, agent i is incentivised to report $\tilde{\epsilon} \geq \epsilon$, where ϵ is the agent's true probability of exceeding for the pair (k, b).*

Proof. Suppose an agent reports a false $\tilde{\epsilon} < \epsilon$, for a bid b that results in an ACCR price p_i. Then, because the agent will exceed their resource limit more often than the protocol expects, it will incur the larger $p_i^\top$ price more often then expected. This results in a price larger than the VCG price p_i^v in expectation, as shown below:

$$\begin{aligned}
E_{\pi_i}[p_i|\tilde{\epsilon}] = & \\
&= (1-\epsilon) \times 0 + \epsilon \times \left(\frac{1}{\tilde{\epsilon}} \cdot p_i^v\right) \\
&= \frac{\epsilon}{\tilde{\epsilon}} \cdot p_i^v \\
&\geq p_i^v \\
&= \frac{\epsilon}{\epsilon} \cdot p_i^v \\
&= (1-\epsilon) \times 0 + \epsilon \times \left(\frac{1}{\epsilon} \cdot p_i^v\right) \\
&= E_{\pi_i}[p_i|\epsilon].
\end{aligned}$$

So agents are equally or better off reporting the true ϵ over a false $\tilde{\epsilon} < \epsilon$. □

Next, we consider if there is an incentive for agents to misreport an $\tilde{\epsilon} > \epsilon$. While reporting an $\tilde{\epsilon}$ too high would make it more likely for a bid to be excluded under (10) or (14), a strategic agent with knowledge of other agents' bids could still manipulate the system. To mitigate this, we take advantage of the fact that ϵ is observable by the auctioneer. This means that after a sufficiently high number of runs, the auctioneer can record and analyse how often agents exceed their allocated resource in comparison to how frequently they say they will. The auctioneer can then disincentivise exaggerating ϵ by charging an extra fee.

Lemma 7. *Given a strategic agent i and a sequence of winning bids $\{B_{i,\alpha}\}$ for agent i over repetitions of the protocol, the auctioneer can verify and punish agents that consistently report $\tilde{\epsilon} > \epsilon$.*

Proof. Because agents are incentivised not to underestimate their probability of exceeding (Lemma 6), the auctioneer can analyse the results of repeated auctions to detect overestimates. We levy an additional fee that arises when the auctioneer is sufficiently confident that an agent has been providing falsified information to the auctioneer, in particular by reporting some $\tilde{\epsilon}$ that is larger than their true probability of exceeding the declared resource use ϵ. After each round, the auctioneer will record the winning bids $\{B_{i,\alpha}\}_i$, along with whether or not an agent exceeded their resource allocation. By treating each auction and resulting executions as a Poisson trial, the auctioneer can determine if an agent has been lying with a sufficiently high confidence.

The auctioneer will consider information from the previous l trials. Suppose agent i was allocated bids with corresponding reported $\tilde{\epsilon}_1, ..., \tilde{\epsilon}_l$ in the previous l auctions, and exceeded their allocated amount of resources e times. Note that their allocated bid in each auction does not have to be identical, nor do they have to correspond to the same implied policies. We consider each execution as a independent Bernoulli variable X_q with mean $\tilde{\epsilon}_q$ representing the boolean variable corresponding to exceeding the resource constraint. We want to know how likely it is that the $X = \sum X_q \leq e$. Let $\mu = \sum \epsilon_q$ be the mean of X. Because we are only concerned when the probability of exceeding is artificially inflated, we to analyse cases where $e < \mu$. Let $\rho = 1 - \frac{e}{\mu}$ be the percent deviation below the mean. Then, using the Chernoff Bounds for the sum of independent random variables corresponding to Poisson trials [Motwani and Raghavan, 1995],

$$\begin{aligned} P[X \leq e] &= P[X \leq (1-\rho)\mu] \\ &\leq e^{-\mu\rho^2/2} \\ &= e^{-\mu(1-\frac{e}{\mu})^2/2} \end{aligned}$$

To achieve a high degree of confidence that an agent is lying, say the auctioneer would only charge a penalty fee on top of the pVCG price to agent i when $e^{-\mu(1-\frac{e}{\mu})^2/2} < \gamma$, where $1 - \gamma$ represents the probability that the agent is lying. □

Taken together, Lemmas 5, 6, and 7 form the proof for the following theorem.

Proposition 8. *The protocol described is incentive compatible for an infinitely repeated auction.*

Proof. A incentive compatible auction occurs when all agents' dominant strategies are to tell the truth, as defined in Section 2.2. In ACCR, there are four ways an agent's strategy might diverge from the truth: (1) by artificially increasing b, (2) by artificially lowering b, (3) by artificially increasing ϵ, and (5) by artificially lowering ϵ. Lemma 5 implies that if agents do not lie via (3) or (4), agents will not lie by (1) or (2). Lemma 6 implies that if agents will not lie via (4). Finally, Lemma 7 implies that for any confidence level, the auctioneer can detect and prevent lying via (3) with a sufficiently high number of trials. Thus, for an infinitely repeated auction, agents dominant strategy is to tell the truth, and thus the protocol is incentive compatible for an infinitely repeated auction. □

4.5 Analysis

In this, section, we complete a runtime analysis of the component parts of the ACCR algorithm.

1. The bid generation algorithm requires one Pareto frontier to be generated for each number of resources $k \leq L$. Approximate Pareto frontier generation is polynomial in the size of the MDP [Forejt et al., 2012, Etessami et al., 2007], which is in this case $|\tilde{\mathcal{M}}_i|$ and thus is $\mathcal{O}(\text{poly}(|\tilde{\mathcal{M}}_i|))$. Because cumulative cost is a state factor in $\tilde{\mathcal{M}}$, the full single-agent bid generation algorithm is $\mathcal{O}(\text{poly}(|\tilde{\mathcal{M}}_i|))$. This computation is decentralised among agents, so the total runtime for bid generation is $\mathcal{O}(\text{poly}(|\tilde{\mathcal{M}}_i|))$.

2. The winner determination problem, i.e., the resource allocation ILP, has a worst-case run time that is exponential in the number of variables and polynomial in the number of constraints [Lenstra, 1983], which in this case is the total number of bids and the number of agents respectively. If we assume each agent submits a fixed number of policies from each Pareto frontier, the

runtime is and thus is $\mathcal{O}(2^{L\times n})$. We note that in practice, ILPs can often be solved via a relaxation to LPs, which can be solved in polynomial time in the number of variables [Wolsey, 2020].

3. Calculating VCG prices for a given agent i requires carrying out a counterfactual winner determination problem without agent i, and thus calculating all prices is n calls to an algorithm that is $\mathcal{O}(2^{L\times n})$. Thus, calculating prices is also $\mathcal{O}(2^{L\times n})$.

We conclude the runtime for the full algorithm is $\mathcal{O}(2^{L\times n} + \text{poly}(|\tilde{\mathcal{M}}_i|))$.

4.6 Evaluation

To evaluate the performance of ACCR, we compared its performance to a series of baselines on the benchmark domain Maze from Wu and Durfee [2010], an advertising budget allocation MDP from Boutilier and Lu [2016], and a real autonomous driving data [Rigter et al., 2022].

4.6.1 Methods

We compared ACCR to four other methods, Model-Checker Multi-Agent Markov Decision Processes (MMDP), Constrained Multi-Agent Markov Decision Processes (CMMDP), Chance-Constrained Column Generation (CG), and Dynamic Chance-Constrained Column Generation (DCG), as described in Section 3.3.1. CMMDP optimise for expected constraints, but the three other baselines optimise for a chance constraint like ACCR. These methods are all cooperative, as we are the first to define the non-cooperative instance of this problem. But because the non-cooperative problem is a harder variant, they remain fair baselines to compare to.

For all methods, LPs and ILPs were implemented with Gurobi, and all MDP methods (e.g., solving the MMDPs, computing maximum reward policies for CG, computing Pareto frontiers) were solved using the PRISM model checker [Kwiatkowska et al., 2002]. All experiments were conducted on an AWS R5a.large

EC2 instance, with 2 CPUs and 16GB of memory. Distributional information on experiments can be found in the Appendix.

4.6.2 The Maze Domain

We first consider the Maze domain described in Section 3.3.1. We choose $L = \frac{hn}{4}$, where h is the global time horizon and n is the number of agents in the system. This means that on average, all agents can use safe actions 25% of the time. This limit was chosen to strike a balance between being higher and thus effectively unconstrained and lower and thus too restrictive for the Column Generation method described below, which uses a very conservative approximation. In all experiments, we bound the probability of resource violations with a chance constraint of $\delta = 0.05$ as in de Nijs et al. [2017]. Each data point in Figure 4.2 and 4.3 represent the average over 50 trials. All methods timeout at 500 seconds.

Results.

In Figure 4.2, we show how the five approaches compare to each other. Note that the environment height matches the environment width, so the number of states in the single-agent model is quadratic in the x-axis. Because the CMMDP is planning for expected resource use and disregarding the chance constraint, it is able to achieve the highest average total reward. However, the constraint is violated between 25%-50% of the time. The MMDP approach provides an optimal solution to the chance-constrained problem, but the planning over the joint model makes it infeasible for large state spaces. The CG approach suffers from the relative value of L_{hoeff} to L. Because L_{hoeff} is 0%-10% of L, even though agents are planning for the expected case instead of the chance constraint, agents have much less latitude to use resources and take risky actions. Even when L_{hoeff} is dynamically increased, it remains far below L in order to guarantee the chance constraint is not exceeded while actually planning for the expected value of L_{hoeff}. In ACCR, the auctioneer has more flexibility to allocate all of the resources. Even as the empirical probability

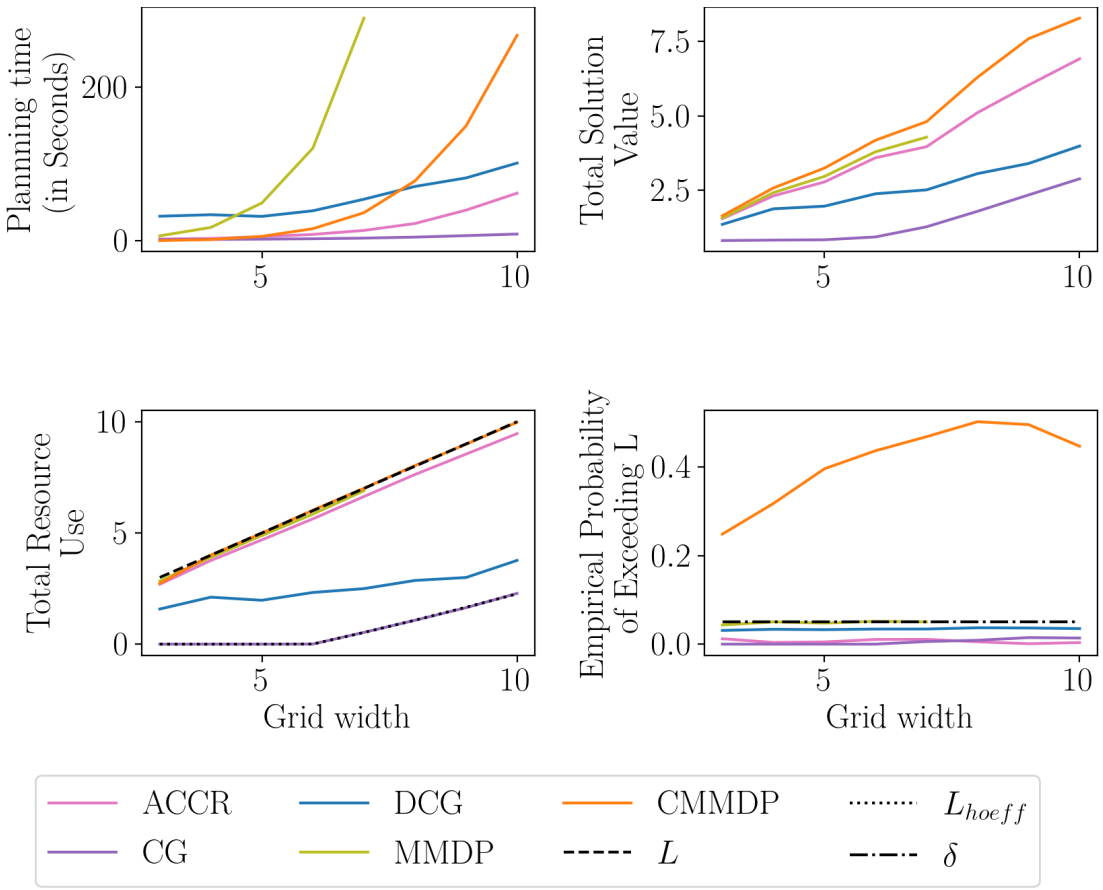


Figure 4.2: Algorithm performance on Maze with increasing state space. Trials are performed with 2 agents. For an analysis of more agents, see Figure 4.3. Data points represent the average over 50 trials. See the Appendix for this chart with error bounds.

of exceeding the resource limit fluctuates, the solution value remains high. ACCR resource utilization is on par with the CMMDP and MMDP approach. However, like the CG approach, all planning over MDPs in ACCR is done on single-agent models, allowing it compute a result more quickly than the joint approaches.

In Figure 4.3, we see how the decentralised natures of ACCR and CG allow for a significant speed up as the number of agents increases, compared to CMMDP. The MMDP approach exceeded the time limit of 500 seconds in all configurations with more than 2 agents, so it is excluded from this experiment. Because L_{hoeff} approaches L as the number of agents increases, both the CG-based approaches achieve a much higher average solution value for problems with more agents. Still ACCR performs better, or the same as, the CG-based models with a large

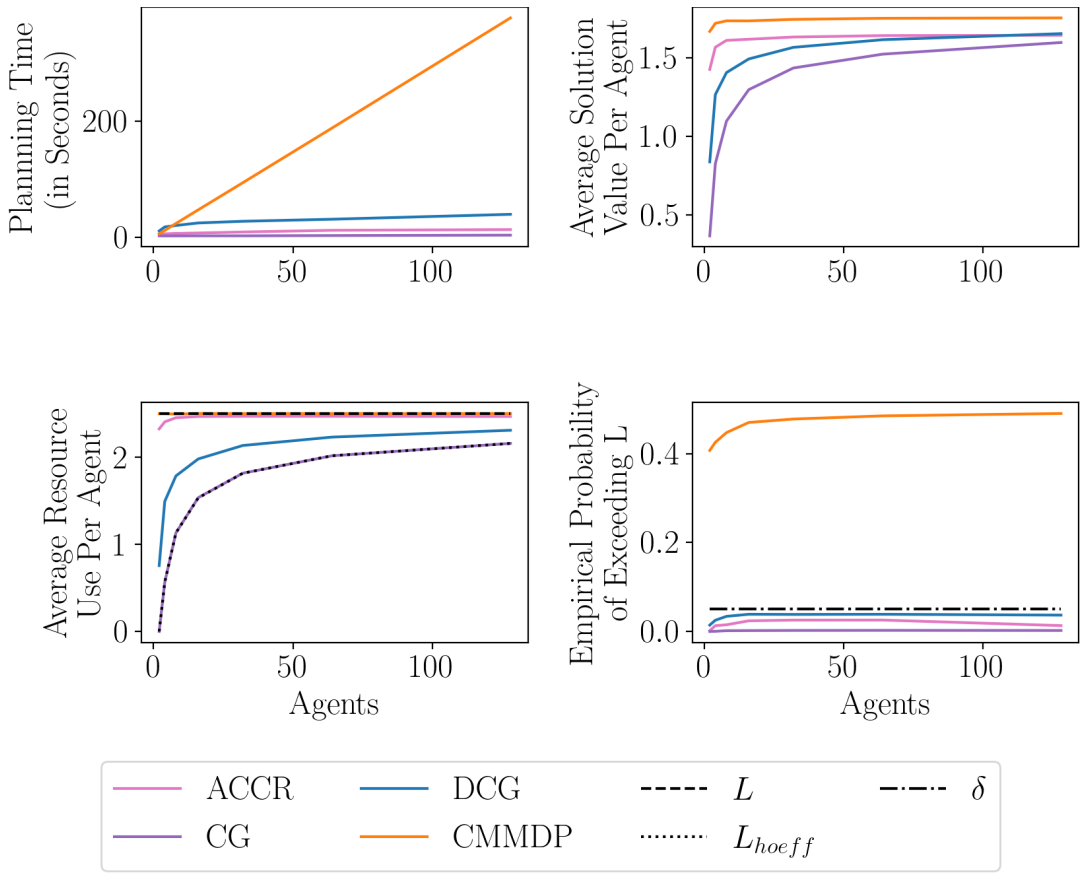


Figure 4.3: Algorithm performance on Maze with increasing agents. Trials were performed with grid width 5. Data points represent the average over 50 trials. See the Appendix for this chart with error bounds.

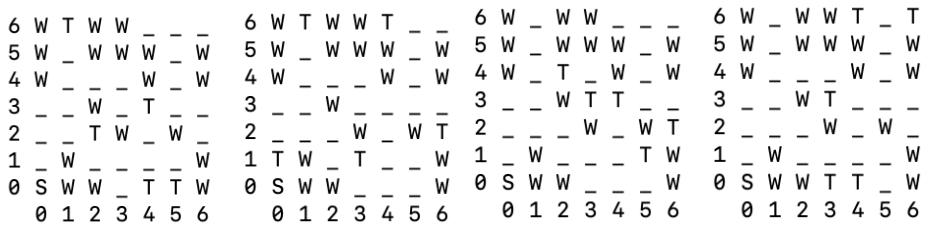


Figure 4.4: Four agent MDP setups on the Maze domain. S corresponds to the start location, T corresponds to task spaces, and W corresponds to walls.

number of agents.

A Closer Look at Policies.

To better investigate the policies returned by each method, we focus on a specific instance of Maze, with grid size 7 and 4 agents with the setups shown in Figure 4.4.

In this setting, $\delta = 0.05$ and $L = 14$.

In Figures 4.5, 4.6, 4.7, and 4.8, we consider the individual agent reward distributions $\mathcal{R}_{i,\pi_i}$ and joint reward distribution $\mathcal{R}_\pi$, individual cost distributions $\mathcal{C}_{i,\pi_i}$ and joint cost distributions $\mathcal{C}_\pi$ for ACCR, CMMDP, CG, and DCG. First, we consider ACCR distributions in Figure 4.5. When examining the individual resource usages, we can see that agents 1 and 2 were allocated resources, where as agents 3 and 4 were not allocated any resources. We can see that agent 1 was allocated 7 resources, and that bid had some positive probability of exceeding those resources, meaning that in a small amount of cases Agent 1 used more than 10 resources. Agent 2, on the other hand was allocated 7 resources and never uses more than 7 resources. Agent 1 and 2 are much more consistently able to reach and execute higher reward tasks because of the resources they received. Where as agents 3 and 4 only occasionally receive higher reward tasks, when they are "lucky" in the outcomes of their no resource actions. When considering the individual agent grids, we gain insight into why Agent 1 and 2 were allocated resources over agent 3 and 4. If we examine agent 4's two most valuable tasks, they are as far as possible for the agents start location. This means that even if agent 4 was allocated more resources to take *safer* actions, it is still more likely that something could go wrong given the distance to travel. In Figure 4.6, we see that the CMDP method instead prioritises resource allocation for Agents 1, 3 and 4. In particular, the reward and cost distributions for Agents 3 and 4 are quite strongly multi-modal. This is because Agents 3 and 4 execute stochastic policies, where around half the time they use 0 resources and half the time they use around 7 and 5 resources respectively. This means that in expectation, the agents jointly use the correct number of resources, but around half the time, the agents use more than the resource limit L. In Figure 4.7, we see that because the CG limit of L_{hoeff} is 0, agents are not permitted to use any resources. Thus, they can only take risky actions. These policies are the result of only taking unsafe actions, and thus frequently fail to successfully execute any task. In Figure 4.8, DCG iteratively relaxes L_{hoeff} until the probability of exceeding the resources limit approaches δ. DCG also considers

stochastic policies, so distributions are also multi modal like in CMMDP. However unlike CMMDP, DCG has a strict limit on the probability of exceeding, so the higher cost mode has a lower probability of occurring than the lower cost mode.

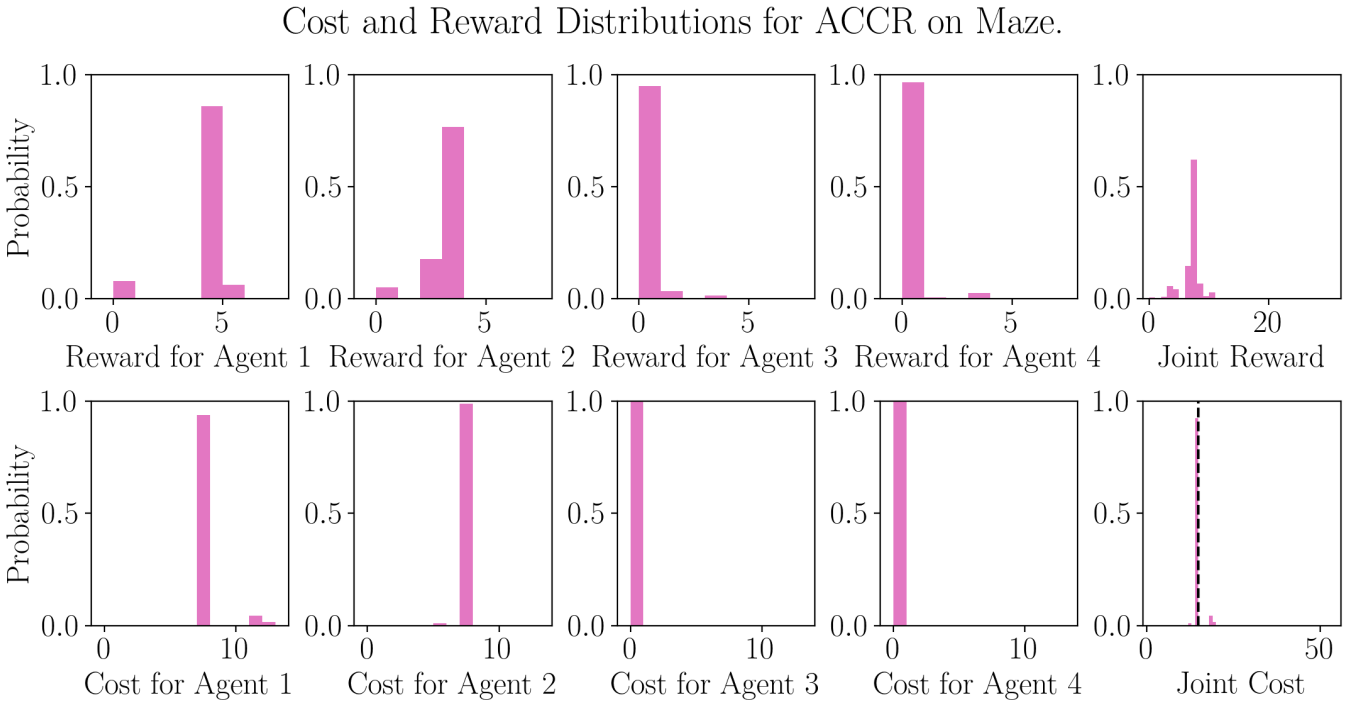


Figure 4.5: From top to bottom and left to right: individual agent reward distributions and joint reward distribution for ACCR; individual cost distributions and joint cost distributions for ACCR. Agent setups are described in Figure 4.4.

4.6.3 The Advertising Domain

Next we consider the Advertising Domain described in Section 3.3.1. As in Boutilier and Lu [2016], the time horizon is 50, though we modify the objective to undiscounted reward. In all experiments, $\delta = 0.05$. For our ACCR algorithm, agents restrict their bids by only generating Pareto frontiers for k divisible by 10. Because the MDP is static, each data point represents a single run of each algorithm.

Results.

Figure 4.9 and Figure 4.10 show that on this domain, CG with dynamic relaxation performs the best, but is much more time consuming than the other algorithms.

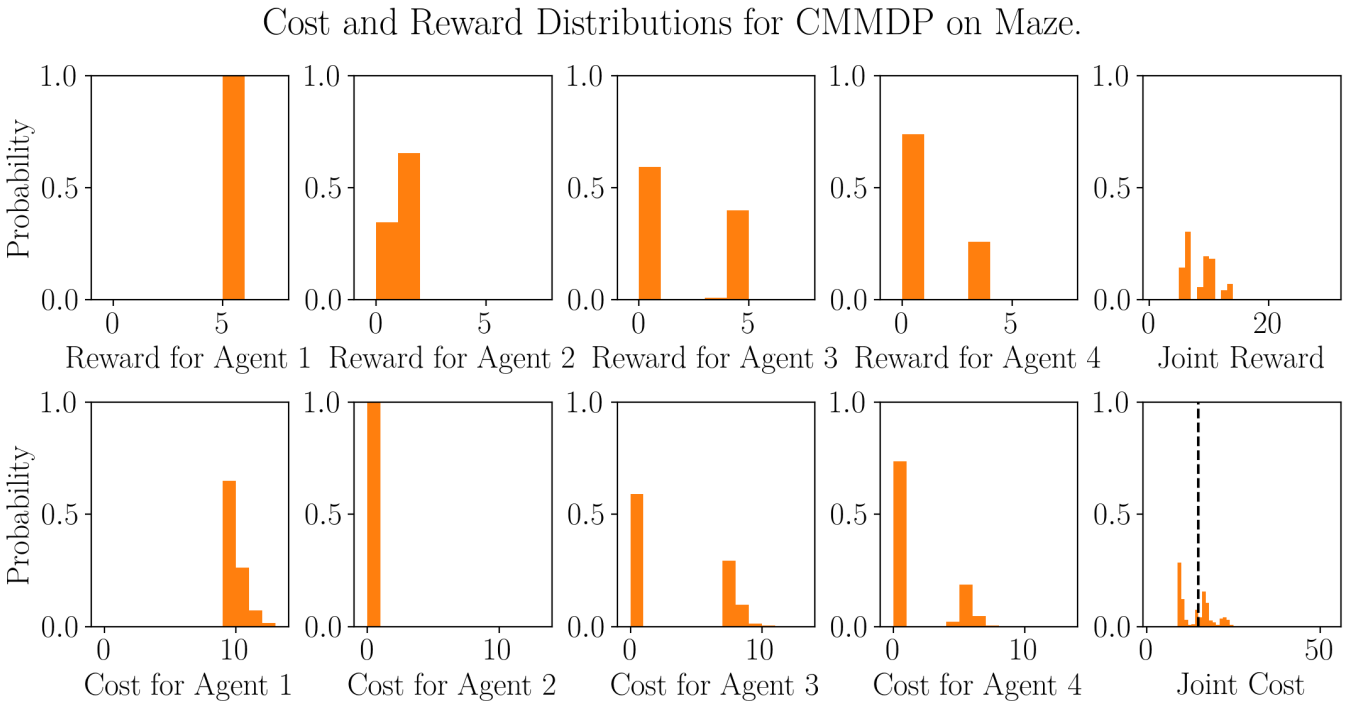


Figure 4.6: From top to bottom and left to right: individual agent reward distributions and joint reward distribution for CMMDP, individual cost distributions and joint cost distributions for CMMDP. Agent setups are described in Figure 4.4.

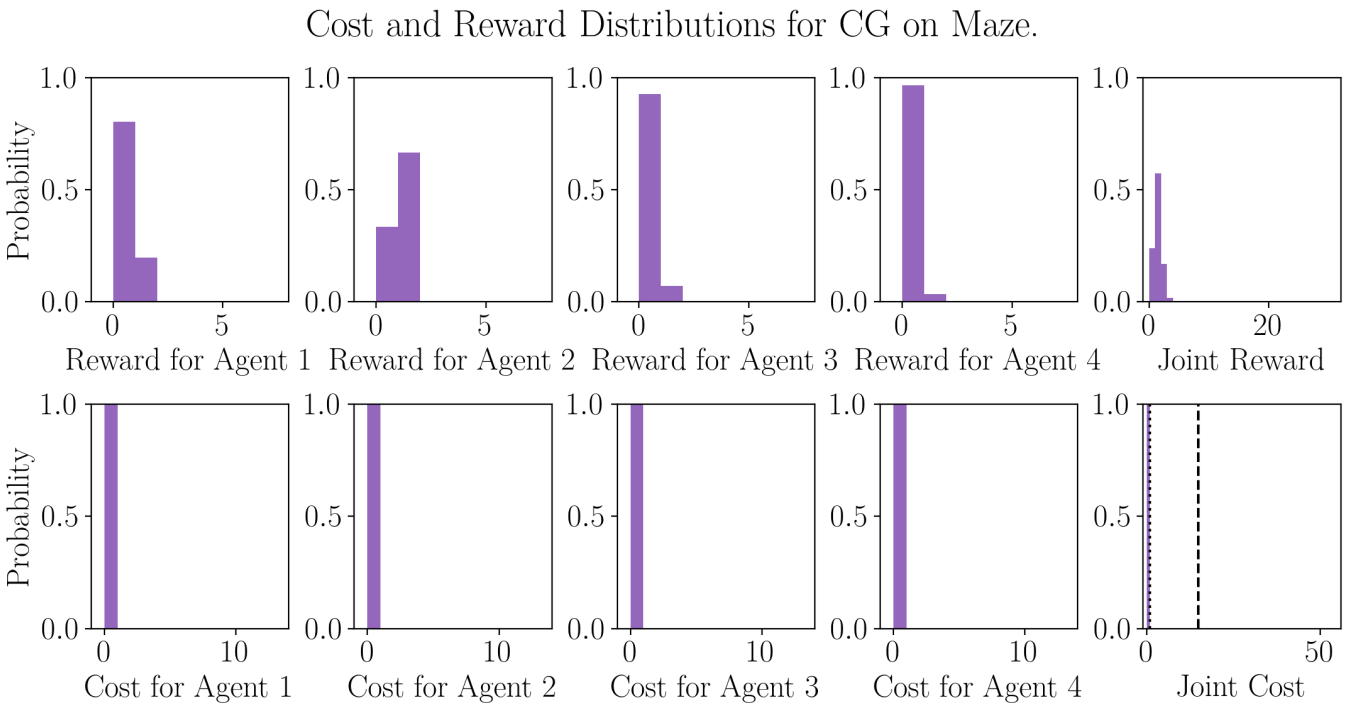


Figure 4.7: From top to bottom and left to right: individual agent reward distributions and joint reward distribution for CG; individual cost distributions and joint cost distributions for CG. Agent setups are described in Figure 4.4.

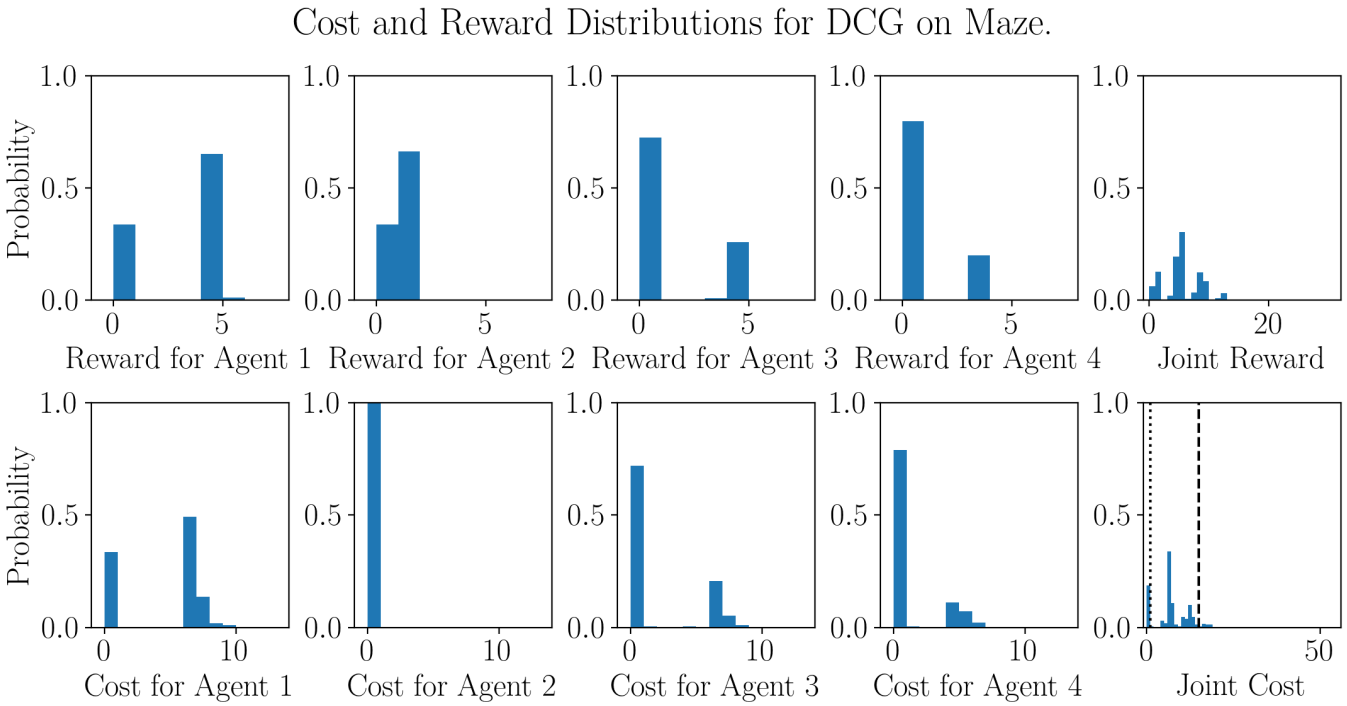


Figure 4.8: From top to bottom and left to right: individual agent reward distributions and joint reward distribution for DCG, individual cost distributions and joint cost distributions for DCG . Agent setups are described in Figure 4.4.

In Figure 4.9, we vary the number of agents and analyse the best way to allocate resources with an average budget of five times the number of agents. We see that ACCR provides a reasonable trade off between the time required by dynamic CG and the performance of non-dynamic CG, particularly for smaller numbers of agents. We also analyse the main limitations of our ACCR algorithm in Figure 4.10, when there are large, variable budgets and large numbers of agents (n=1000). The CG algorithms uses the law of large numbers to plan for the expected case, so they perform best with large numbers of agents and loose constraints. The ACCR algorithm is outperformed on this domain instance because the use of deterministic policies and the limited bidding prevent the algorithm from taking advantage of all the resources available.

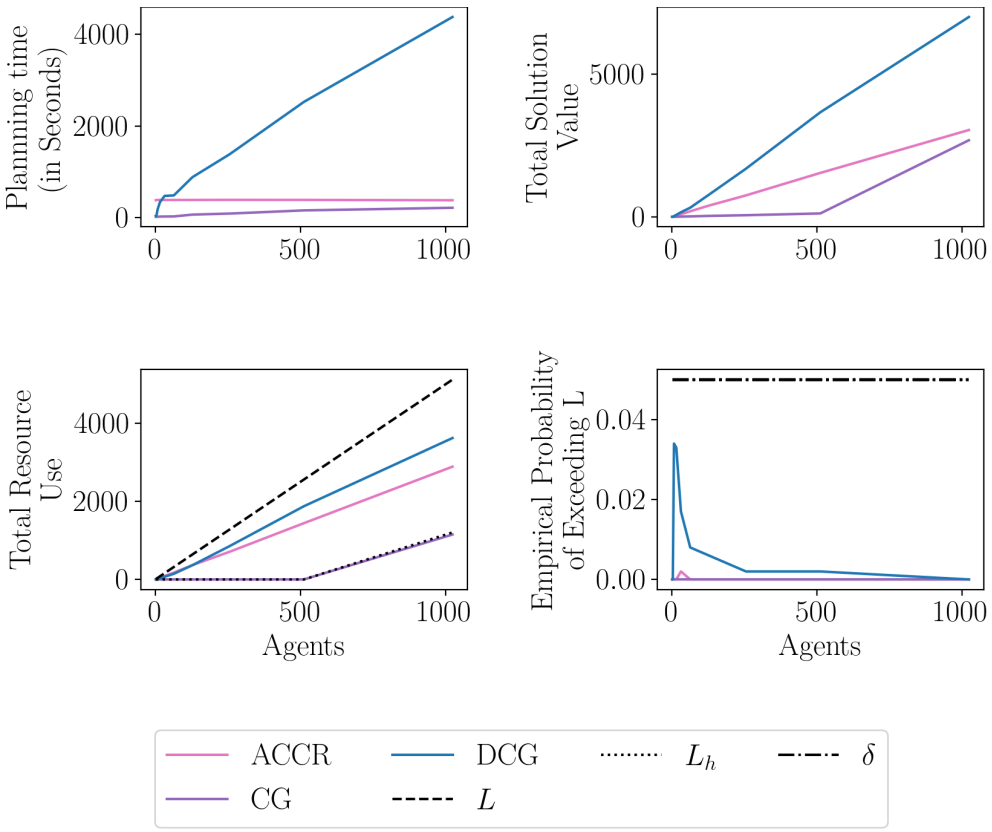


Figure 4.9: Algorithm performance on Synthetic Advertising Domain with increasing numbers of agents.

4.6.4 An Autonomous Driving Network

Finally, we consider the autonomous driving domain as described in Section 3.3.1 As in the other experiments, $\delta = 0.05$ and L is $\frac{1}{4}$ the maximum resource usage. For our ACCR algorithm, agents restrict their bids by only generating Pareto frontiers for k divisible by 20.

Results.

In Figure 4.11 we tested the performance of Maze over 50 trials with up to 6 agents. In this setting, the high-variability of costs leads to longer planning time; there are more possible Pareto fronts that need to be evaluated, and the size of the extended single-agent model ($|\tilde{\mathcal{M}}_i|$) is much larger. Some agents, though are able to decrease their planning time by ceasing to generate Pareto fronts when their expected reward

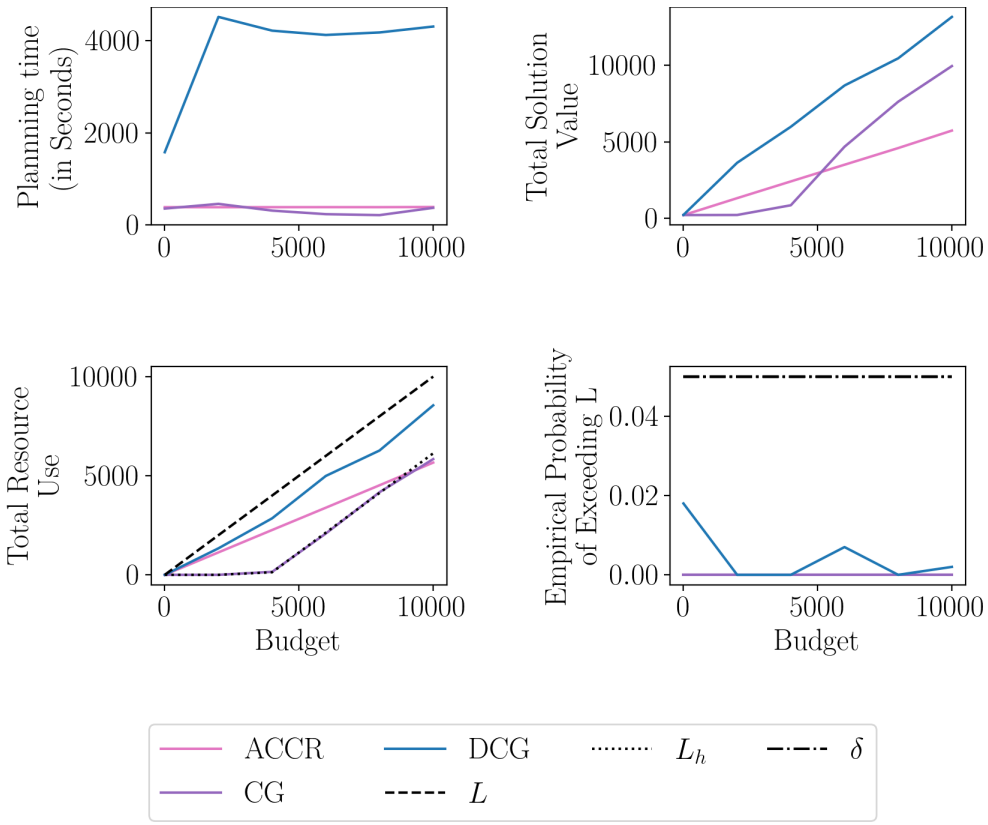


Figure 4.10: Algorithm performance on Synthetic Advertising Domain with increasing budgets.

for a given k is identical for both $\epsilon = 1$ and $\epsilon = 0$. This only occurs when there is no possible policy option for the agent to take that will ever exceed k resources. This bidding behaviour also explains why the planning time increases so drastically with the number of agents; the time for the auction remains similar, but the more agents that are in the auction, the more likely there is an agent that has to reach a larger k until the reward is identical for both $\epsilon = 1$ and $\epsilon = 0$, and thus the maximum single agent bid generation time is more likely to take longer. This also explains why the average bid use is consistently less than the resource limit, in most configurations, the agents cannot actually use all the available resources. We can instead examine a specific configuration, where agents are able to achieve the resource limit.

In Figure 4.12, we consider a specific instance of the autonomous driving domain. In this instance, we ACCR took 2366 seconds to run, achieved a reward of 592.7, at a average cost of 309 travel time. In this case, the agents exceeded the resource

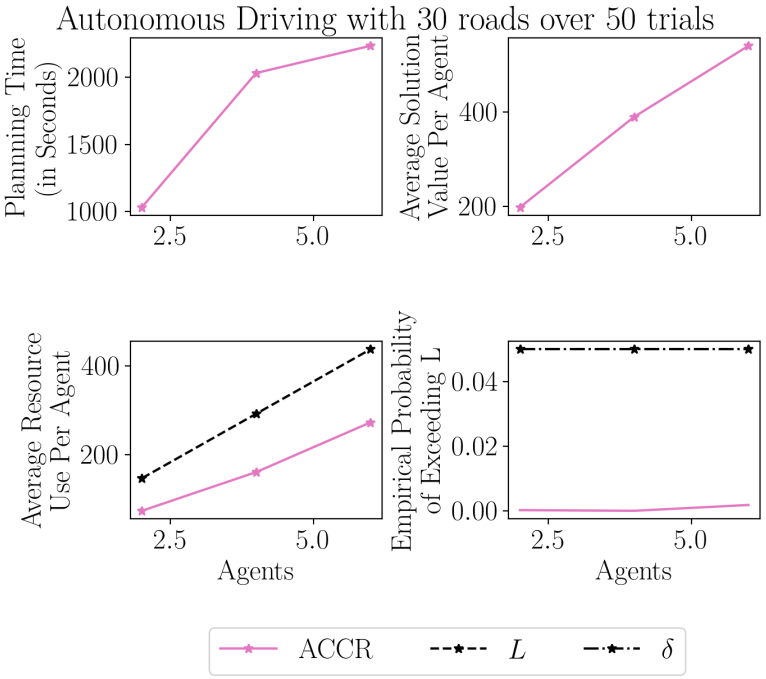


Figure 4.11: ACCR performance on the autonomous driving domain from Rigter et al. [2022] with 30 roads.

limit with probability 0.009.

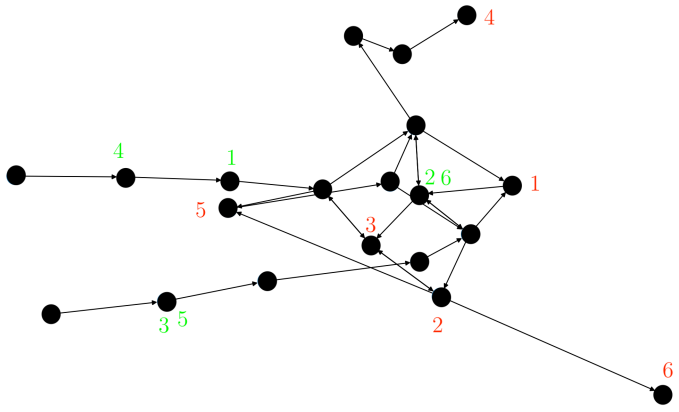


Figure 4.12: An instantiation of the autonomous driving domain from Rigter et al. [2022] with 30 roads and 6 agents.

4.7 Discussion

4.7.1 Summary

In the Chapter, we presented ACCR, an auction-based mechanism for allocating chance-constrained, multi-unit resources. By designing an auction to include information about agents' uncertainty over resource use, ACCR can effectively allocate resources, while at the same time limiting resource violations. We demonstrated an efficient implementation for ACCR with an ILP (Section 4.2). We also presented an algorithm for individual agents that uses off-the-shelf tools to generate bids (Section 4.3). We discussed how ACCR can be applied to non-cooperative scenarios with prices (Section 4.4). Finally, we empirically showed that ACCR outperforms other chance-constrained methods on the Maze benchmark (Section 4.6).

4.7.2 Limitations and Future Work

In cooperative settings, ACCR is the best existing method to solve the Chance-Constrained Multi-Agent Planning Problem defined in Equation 4.2 for small to moderate number of agents. We allow for decentralised planning and centralised resource use handling through a novel auction formulation. In the cooperative setting, Chance-Constrained Column Generation is a better alternative over ACCR when the numbers of agents is a large enough to sufficiently tighten the Hoeffding bound. This is because the use of the column generation algorithm allows for a more efficient choice on what policy options to generate. Alternatives to ACCR, like Chance-Constrained Column Generation, are also applicable to instantaneous resources where as ACCR is only suitable to budget constrained resources. In theory, ACCR can be extended to instantaneous resources. Doing so would require bidding to include a dimension of the Pareto frontier for each timestep. Similarly, ACCR can be extended to multiple resources by adding a dimension to the Pareto frontier for each resource. In fact, the instantaneous resource can can be seen as a special case of multiple resources. Unfortunately, approximate Pareto Curve is at best polynomial in the number of dimensions [Papadimitriou and Yannakakis, 2000] and

at worst exponential in the number of dimensions [Etessami et al., 2007], and the number of calls to create a Pareto frontier will be $O(L_1 \times L_2 \times \cdots \times L_m)$ where L_z is the resource limit from each resource z. One interesting area of future work would be to intelligently search the space of possible Pareto frontiers to limit computation. Another possible area of research would be an iterative auction like SSI [Lagoudakis et al., 2005], which would allow agents to iteratively bid on multiple resources.

Another limitation of ACCR compared to Column Generation is the deterministic nature of the ACCR policies. ACCR's initial INLP can be easily modified to synthesise stochastic policies; the integral variables in Equation 4.8 must be replaced with a continuous variables. But, this modification has two major downsides. First, the exact transformation from a INLP to a ILP is no longer valid. An important step for future work would be developing an exact or approximate ILP under the new substitution. Second, the incentives of stochastic policies need to be considered.

In non-cooperative settings, ACCR remains the only method for chance-constrained resource allocation. However, ACCR is not suitable for every non-cooperative scenarios due to lie detection mechanism. One-off auctions would not have enough data to detect liars, so a series of repeated auctions with the same agents are required. Note that all agents do not have to be the same in every auction, just that each agent participates in multiple auctions. Because of the confidence γ on the lie detection method, it is possible that an agent who did not lie and just got very lucky would be penalised. Decreasing γ can control how lucky an agent would need to be to be unfairly punished, but this comes at the cost of catching some lying agents. As the confidence level reaches 1, the number of trials required tends to infinity, so this auction should be used in scenarios where complete confidence is not required.

The final point of discussion is on the problem formulation, chance-constrained resource allocation. Chance-constrained resource allocation focuses on optimising the majority of scenarios (say the top 95%) and as a result ignores the worst 5% of case. In many settings this is sufficient. For example, you may have a human operator who can handle the worst cases manually. Or you may be in a scenario where the worst cases are not safety critical. Imagine a team of self driving delivery

robots. If the delivery robot exceed their time limit for delivery in a small percent of cases, it may be an expected downside of operations. But there are safety critical settings where it is important to consider what happens in the worst case. Imagine a doctor is synthesising a policy to administer a radiation treatment for cancer. Their optimisation metric is the success probability of curing cancer, but their resource constraint is limiting the amount of radiation that the good cells are exposed to. In such cases, it would be necessary to consider the maximum radiation that the person is exposed to, instead of ignoring the worst cases. In the less safety critical domain of the advertising budget domain, where money is involved, the worst cases may need to be considered as well. For example, imagine if we ignore that 5% worst cases, but the optimal policy has a cost distribution that results in agents jointly spending £1,000,000,000 in 4% of cases. This would (likely) be unacceptable. This could be handled by required human intervention after a certain spend limit, but if we want to handle this fully autonomously, we need an optimisation metric that has richer understanding of what happens in the worst case. We will explore one such method to do this in the Chapter 5.

5

Allocating Risk-Constrained Resources

5.1 Introduction

In this chapter, we consider the same setting as Chapter 4, resource allocation of multi-unit resources under uncertainty, but we instead consider a more complex notion of safe resource usage, namely Conditional Value at Risk (CVaR). CVaR considers the average resource value within the worst case tail of a distribution. In Chapter 4, we considered any run that exceeded the resource limit a failure, and our optimisation metric sought to limit such failures. In this chapter, we consider the distribution of *the set of the worst cases*, and seek to limit the average resource use in that set. This allows us to have a more complete understanding of resource usage in the extremes of our distribution, and allows us to be robust in the worst cases. Because a CVaR constraint is a richer quantitation of uncertainty than a chance constraint, it requires more complex algorithms. In particular, CVaR cannot be neatly decomposed into individual contributions that agents can calculate without considering the presence of other agents. Contrast this with Chapter 4, where agents independently generated their *probability of exceeding* some number of resources and these could be easily collated into the joint probability of exceeding the global resource constraint. With CVaR, how much risk an agent contributes, i.e. the resource they use in the worst case tail, is dependent where the worst-case

Table 5.1: A description of Chapter 5's relevance to the three main challenges of multi-agent decision making.

	Chapter 4	Chapter 5	Chapter 6
Uncertain Domain?	Yes	Yes	No
Rich Uncertainty?	No	Yes	No
Multiple Resource Types?	No	No	Yes
Non-Cooperative?	Yes	No	Yes

tail is, which is in turn dependent on the other agents' resource usage. Thus, this notion of resource constraints is ill-suited to auction based methods because the joint CVaR for a set of agents is not easily decomposed into individual contributions. As a result, in this chapter we explore a specific method of *attribution* called risk contributions, and explore how an agents contribution to the joint risk is dependent on the other agents resource usage. In particular, an agents risk contribution is dependent on the full distributional information of other agents' resource usages. In Section 2.1.5 we will formally define CVaR and risk contributions, and discuss the important dependencies between them. Then we use the risk contribution method of attribution to address the problem of multi-agent risk-constrained planning, which we define as the problem of maximising for joint reward while constraining the CVaR of joint cost in a multi-agent system in Section 5.2.1. Our main contributions are:

1. formally defining the risk-constrained, multi-agent planning problem ;
2. developing a method of approximating an agent's risk contribution with only the summary data of other agents' distributions; and
3. presenting an algorithm that solves the risk-constrained, multi-agent planning problem by using our approximate *risk contribution* to decompose a multi-agent planning problem into a series of single-agent planning problems.

This is, to the best of our knowledge, the first work to use risk contributions out of quantitative finance in the context of multi-agent decision making under uncertainty. Using the notion of risk contribution, our algorithm identifies agents who contribute proportionally more risk and incrementally updates their policies

(Section 5.2.3). Policy updates are carried out using a modification of the single-agent risk-constrained planning problem from Borkar and Jain [2014] (Section 5.2.4). Finally, empirical analysis shows that this substantially improves scalability, allowing us to solve problem instances that are out of reach for the current state-of-the-art of planning over the joint model. Given the novelty of the problem addressed here, we focus on the problem of cooperative planning, and leave the development of non-cooperative planning to future work (Section 5.3).

This chapter describes the work published in Gautier et al. [2023b] in the Proceedings of AAMAS 2023.

5.2 Risk-Constrained Planning

5.2.1 Problem Description

Now we introduce the problem of risk-constrained multi-agent planning.

Definition 31 (Risk-Constrained Multi-Agent MDP)**.** *A Risk-Constrained Multi-Agent MDP (RCMMDP) is a triple* $\langle \mathcal{M}, L, \delta \rangle$*.* $\mathcal{M}$ *denotes the weakly-coupled MMDP.* $\delta \in (0, 1]$ *denotes the size of the tail of the distribution.* $L \in \mathbb{N}^+$ *denotes the limit of the globally constrained budget resource – which is to be constrained by CVaR:*

$$CVaR_\delta \left[\sum_{i \in [n]} \mathcal{C}_{i,\pi_i} \right] < L. \tag{5.1}$$

The goal of the RCMMDP is to find a joint policy $\pi^\star = \{\pi_i^\star\}_{i \in [n]}$ that maximises the cumulative reward, subject to the risk constraint:

$$\pi^\star = \underset{\pi := \{\pi_i\}_{i \in [n]}}{\arg\max} \quad E_\pi \left[\sum_{i \in [n]} \mathcal{R}_{i,\pi_i} \right], \tag{5.2}$$

$$\text{s.t.} \quad \text{CVaR}_\delta \left[\sum_{i \in [n]} \mathcal{C}_{i,\pi_i} \right] < L. \tag{5.3}$$

Example 5. *Recall the problem of running an advertising campaign over time through 1000 automated agents, as described in Chapter 4. Each agent is in charge of planning one personalised campaign, which attempts to get a single consumer to*

purchase their product. This problem can be modelled as a multi-agent sequential decision making problem under uncertainty: for each agent 15 states represent different interest levels of their customer, ranging from uninterested to purchasing a product. When a customer reaches the state of buying a product, the agent gets a reward. Towards this purpose, each agent has 5 action choices to make about what marketing strategy to employ at each time based on their customer's current interest level. Action choices which are more effective at moving the customer toward purchasing the agent's product are also most monetarily costly. The goal of the firm would be to convert as many potential customers to paying customers as possible. But there is also a monetary constraint on how much money the agents can spend jointly on converting customers which needs to be handled. Because the same action could result in a different future interest level, which may in turn require a different cost action, the monetary cost for any given advertising strategy results in a distribution over possible final monetary costs. In this setting, the risk-constrained multi-agent planning problem asks: What strategy should each automated agent take, such that as many customers as possible are converted to purchase the project, subject to the constraint that in the worst 5% *of cases the money spent is on average less than* $100.

5.2.2 A Naive Approach

The naive approach to solving this problem would be to treat $\mathcal{M}$ as a strongly-coupled MMDP and directly apply the single-agent algorithm iRMDP to $\mathcal{M}$. This approach has two downsides. First, the planning via iRMDP happens over the joint state and action space. This can be prohibitively time consuming, particularly for large numbers of agents. This is exacerbated by the fact that iRMDP runs value iteration on the state space *and* the cost accumulated so far. Having to consider the joint cumulative cost further expands the computation that is needed. Second, the polices returned will also be strongly-coupled, meaning that agents' actions will depend on the joint state space. This means that when agents execute their

offline polices, agents must be able to communicate, or have full observability over the joint state space.

5.2.3 A Risk Contribution Approach

To solve the RCMMDP problem defined in Equations (5.2-5.3), we present the Risk Contribution Approach (RCA) algorithm. RCA is an iterative algorithm that starts with a set of single-agent policies that optimise solely for reward, and then iteratively updates one policy at a time based on the agents' risk contributions, until a set of single-agent policies that satisfies Equation (5.3) is synthesised. We define the iterative updates and describe how single agents update their policy. This approach is described in Algorithm 2.

Algorithm 2 RCA: An approximate RCMMDP Planner

Require: an MMDP $\mathcal{M} = \mathcal{M}_1 \times \cdots \times \mathcal{M}_n$, an intial state s_0, a confidence level $\delta \in (0, 1]$, and a risk-bound $L \in \mathbb{N}^+$, a stepsize $\gamma \in \mathbb{N}^+$

1: **for** $i \in [n]$ **do**
2: $\quad \pi_i = \text{RiskNeutralPolicy}(\mathcal{M}_i)$
3: **end for**
4: $var_0, cvar_0, \{rc_{0,i}\}_{i\in[n]} = \text{CalcRisk}(\mathcal{M}, \{\pi_i\}_{i\in[n]}, \delta)$
5: **for** u=1,2,...,until $cvar_{u-1} < L$ **do**
6: $\quad j = \arg\max\{\frac{rc_{u-1,i}}{E[\mathcal{R}_{i,\pi_i}]} | i \in [n]\}$
7: $\quad \tilde{rc}_j = rc_{u-1,j} - \gamma$
8: $\quad \tilde{\beta} = var_{u-1} - \sum_{i\neq j} rc_{u-1,i}$
9: $\quad \pi_j = \text{RCConstrainedPolicy}(\mathcal{M}_j, \tilde{rc}_j, \tilde{\beta})$
10: $\quad var_u, cvar_u, \{rc_{u,i}\}_{i\in[n]} = \text{CalcRisk}(\mathcal{M}, \{\pi_i\}_{i\in[n]}, \delta)$
11: **end for**
12: **return** $\{\pi_i\}_{i\in[n]}$

Lines 1-3 initialise policies for each agent. These initial policies are risk neutral, meaning they maximise for the reward function (R_i) without taking into account the cost function (C_i). These can be found with any single-agent MDP solver, e.g., value iteration. We begin with risk neutral policy to gain information about agents risk contributions when they planned in an unconstrained manner. Line 4 calculates the risk associated with the initial policies using Monte Carlo trials. This yields the VaR of the *joint* cost, $VaR_\delta(\sum_{i\in[n]} \mathcal{C}_{i,\pi_i})$, the CVaR of the *joint* cost,

$CVaR_\delta(\sum_{i\in[n]} \mathcal{C}_{i,\pi_i})$, and each agent's risk contribution $\text{RC}_{\delta,i}$. An implementation of the risk calculation in line 4 based on rejection sampling is detailed in Section 5.3.

Then, we proceed to the main part of the algorithm, which chooses an agent and iteratively updates that agent's policy until the global constraint is met.

First, in line 5 we check if the CVaR of the current set of policies meets the risk-bound L. If so, we continue to the end.

Line 6 identifies the agent j with the worst reward-to-risk trade-off, by comparing each agent's risk contribution with their current policy to their expected reward with that policy. We then set agent j with a new risk contribution goal. This seeks to reduce agent j's current risk contribution, $rc_{u,j}$, by some step size γ, to $\tilde{rc_j} := rc_{u-1,j} - \gamma$ (line 7).

By definition, the optimal best response reduction in agent j's risk contribution (to $\tilde{rc_j}$) is to find a new policy π_j which optimises for reward ($E_{\pi_j}[\mathcal{R}_{j,\pi_j}]$) while constraining by:

$$E_\pi \left[\mathcal{C}_{j,\pi_j} | \sum_{i\in[n]} \mathcal{C}_{i,\pi_i} \geq VaR_\delta(\sum_{i\in[n]} \mathcal{C}_{i,\pi_i}) \right] \leq \tilde{rc_j}. \tag{5.4}$$

Exactly optimising for Equation 5.4 would still require reasoning over the joint state space, as the joint resource use would be necessary to successfully calculate which outcomes are counted within the conditional expectation. To avoid reasoning over the joint state space, we approximate the distributional information that corresponds to the other agents' state spaces and action choices under their policies $\{\pi_i\}_{i\neq j}$.

First, we approximate the true VaR ($\text{VaR}_\delta(\sum_{i\in[n]} \mathcal{C}_{i,\pi_i})$) with the VaR from the previous iteration's set of policies (denoted by var_{u-1}). This yields:

$$E_\pi \left[\mathcal{C}_{j,\pi_j} | \sum_{i\in[n]} \mathcal{C}_{i,\pi_i} \geq var_{u-1} \right] \leq \tilde{rc_j}. \tag{5.5}$$

We argue this is a reasonable assumption given a small choice of step size: consider the optimal policy $\pi_j^\star$ (with respect to satisfying Equation 5.4). $\text{VaR}_\delta(\sum_{i\in[n]} \mathcal{C}_{i,\pi_i})$ will be similar to $\text{VaR}_\delta(\mathcal{C}_{j,\pi_j^\star} + \sum_{i\neq j} \mathcal{C}_{i,\pi_i})$ because $\{\pi_i\}_{i\neq j}$ remain consistent and $\pi_j^\star$ and π_j are are the optimal policies for only slightly different optimisation criteria.

Next, we approximate the other agents' resource usages in the left-hand side of the constraint:

$$E_\pi[\mathcal{C}_{j,\pi_j} | \sum_{i\in[n]} \mathcal{C}_{i,\pi_i} \geq var_{u-1}] \tag{5.6}$$
$$= E_\pi[\mathcal{C}_{j,\pi_j} | \mathcal{C}_{j,\pi_j} + \sum_{i\neq j} \mathcal{C}_{i,\pi_i} \geq var_{u-1}] \tag{5.7}$$
$$\approx E_\pi[\mathcal{C}_{j,\pi_j} | \mathcal{C}_{j,\pi_j} + \sum_{i\neq j} E_{\pi_i}[\mathcal{C}_{i,\pi_i} | \sum_{i\in[n]} \mathcal{C}_{i,\pi_i} \geq var_{u-1}] \geq var_{u-1}] \tag{5.8}$$
$$= E_{\pi_j}[\mathcal{C}_{j,\pi_j} | \mathcal{C}_{j,\pi_j} \geq var_{u-1} - \sum_{i\neq j} E_{\pi_i}[\mathcal{C}_{i,\pi_i} | \sum_{i\in[n]} \mathcal{C}_{i,\pi_i} \geq var_{u-1}]] \tag{5.9}$$

First at Line 5.7, we separate out agent j's contribution to the cost ($\mathcal{C}_{j,\pi_j}$) from rest of the joint cost ($\sum_{i\neq j} \mathcal{C}_{i,\pi_i}$). Then at Line 5.8, we remove the dependence on the full random variables ($\sum_{i\neq j} \mathcal{C}_{i,\pi_i}$) by instead taking the average value of each random variables for $i \neq j$ under the conditional ($\mathcal{C}_{i,\pi_i} \geq var_{u-1}$). Finally, at Line 5.9 we note that $\sum_{i\neq j} E_{\pi_i}[\mathcal{C}_{i,\pi_i} | \sum_{i\in[n]} \mathcal{C}_{i,\pi_i} \geq var_{u-1}]$ is constant, and thus (1) the expectation is only over π_j instead of π and (2) we can move $\sum_{i\neq j} E_{\pi_i}[\mathcal{C}_{i,\pi_i} | \sum_{i\in[n]} \mathcal{C}_{i,\pi_i} \geq var_{u-1}]$ to the right-hand side of the conditional.

Because $E_{\pi_i}[\mathcal{C}_{i,\pi_i} | \sum_{i\in[n]} \mathcal{C}_{i,\pi_i} \geq var_{u-1}] = rc_{u-1,i}$, this is results in our final constraint:

$$E_{\pi_j}\left[\mathcal{C}_{j,\pi_j} | \mathcal{C}_{j,\pi_j} \geq var_{u-1} - \sum_{i\neq j} rc_{u-1,i}\right] \leq r\tilde{c}_j. \tag{5.10}$$

Using this approximation we set $\tilde{\beta} := var_{u-1} - \sum_{i\neq j} rc_{u-1,i}$, which becomes the point on the x-axis of agent j's distribution at which outcomes are considered part of their risk contribution. This approximation allows us to plan only on agent j's MDP as it represents the actions of the other agents in the system as constants.

Then, in line 9 we find a policy π_j that satisfies the new risk contribution constraint (i.e., maximising for reward while satisfying Constraint 5.10) with RCConstrainedPolicy, described in Section 5.2.4, which is a modification of the single-agent iRMDP. Finally, line 10 calculates the risk associated with the current policies in the same way as line 4. This process is repeated until the risk constraint is met, at which point the current policies are returned.

5.2.4 Risk-Contribution-Constrained Policy Planner

The RCConstrainedPolicy algorithm (Algorithm 3) solves the single-agent risk-contribution reduction problem. Given a desired risk-contribution $r\tilde{c}_j$ and a quantile $\tilde{\beta}$, the RCConstrainedPolicy algorithm optimises for a risk-contribution constraint:

$$\pi_j^\star = \arg\max_{\pi_j} \quad E_{\pi_j}\left[\mathcal{R}_{j,\pi_j}\right], \tag{5.11}$$

$$\text{s.t.} \quad E_{\pi_j}\left[\mathcal{C}_{j,\pi_j} | \mathcal{C}_{j,\pi_j} \geq \tilde{\beta}\right] \leq r\tilde{c}_j. \tag{5.12}$$

Algorithm 3 accomplishes this by modifying the iRMDP described in Section 3.3.1, and those modifications are noted with a $\star$.

Algorithm 3 RCConstrainedPolicy

Require: A single-agent MDP $\mathcal{M}_i := \langle S_i, A_i, T_i, R_i, C_i, h\rangle$, a risk contribution limit $r\tilde{c}_j$, a tail bound $\tilde{\beta}$.

$\lambda^0 = 0$
for w=1,2,...,until converged **do**
  $\delta^{w,0}$ near 0 ▷ $\star$
  for v=1,2,...,until converged **do**
    $J_{h+1}[s,y] = \lambda^w(r\tilde{c}_j - \frac{y}{\delta^{w,v}}\mathbb{1}_{(y>\tilde{\beta})})$ ▷ $\star$
    $V_{h+1}[s,y] = \mathbb{1}_{(y>\tilde{\beta})}$ ▷ $\star$
    $Q_{h+1}[s,y] = \frac{y}{\delta^{w,v}}\mathbb{1}_{(y>\tilde{\beta})}$ ▷ $\star$
    for $t = h, h-1, ..., 0$ **do**
      $G_t^{w,v}[s,y,a] = R_i(s,a) + \sum_{s'} T_i(s,a,s')J_{t+1}^{w,v}[s', y + C_i(s,a)]$
      $J_t^{w,v}[s,y] = \max_{a\in A} G_t^{w,v}[s,y,a]$
      $\pi_t^{w,v}[s,y] = \arg\max_{a\in A} G_t^{w,v}[s,y,a]$
      $V_t^{w,v}[s,y] = \sum_{s'} T_i(s, \pi_t^{w,v}[s,y], s')V_{t+1}^{w,v}[s', y + C_i(s, \pi_t^{w,v}[s,y])]$
      $Q_t^{w,v}[s,y] = \frac{C_i(s,\pi_t^{w,v}[s,y])V_t^{w,v}[s,y]}{\delta^{w,v}} + \sum_{s'} T_i(s, \pi_t^{w,v}[s,y], s')Q_{t+1}^{w,v}[s', y + C_i(s, \pi_t^{w,v}[s,y])]$ ▷ $\star$
    end for
    $\delta^{w,v+1} = \delta^{w,v} - \frac{1}{v}(\delta^{w,v} - V_0^{w,v}[s_0, 0])$ ▷ $\star$
  end for
  $\lambda^{w+1} = (\lambda^w - \frac{1}{w}(r\tilde{c}_j - Q_0^{w,v}[s_0,0]))^+$
end for

Like iRMDP we solve the Lagrangian relaxation:

$$\min_{\lambda\geq 0}\max_{\pi_j} E_{\pi_j}\left[\mathcal{R}_{j,\pi_j}\right] + \lambda\left[r\tilde{c}_j - E_{\pi_j}\left[\mathcal{C}_{j,\pi_j} | \mathcal{C}_{j,\pi_j} \geq \tilde{\beta}\right]\right], \tag{5.13}$$

Recall that in iRMDP, in order to calculate and bound CVaR, δ (i.e the percent of the tail to evaluate) is given as an input, the outer loop iterates over λ, the middle loop iterates over β (i.e, the value at which the tail begins), and the inner loop conducts value iteration and policy evaluation.

In our case, the goal is to calculate a bound the risk contribution and so $\tilde{\beta}$ (i.e., the joint value at which the tail begins) is given as an input. Since we also need to solve a Lagrangian relaxation, the outer loop (lines 2-18) still iterates over λ. Unlike in iRMDP, the middle loop (lines 3-16) instead needs to iterate over δ (i.e, the percent of the tail to evaluate) in order to successfully calculate Q via policy evaluation. Note that the initial condition of δ, $\delta^{w,0}$ needs to be some real number sufficiently close to 0, but not 0, to avoid division by 0 in line 7. Then, like in iRMDP the inner loop (8-14) conducts value iteration and policy evaluation.

5.3 Evaluation

We evaluated the performance of our multi-agent algorithm against iRMDP on two benchmark domains: Maze and advertising budgets from Chapter 4.

5.3.1 Methods

We compared RCA to three other methods, iRMDP, our Auction for Chance-Constrained Resources, and a Risk Neutral Policy.

1. The **iRMDP** method is as described in Algorithm 1. To do this, we treat the weakly-coupled MMDP as a strongly-coupled MMDP, which can then be treated as any other MDP by iRMDP. The convergence condition for both loops is set to .01.

2. The **Auction for Chance Constrained Resources** (ACCR) method is as described in Chapter 4, with a centralised planning time.

3. The **Risk Neutral** (RN) method only optimises for reward. Policy synthesis for this method is done with the PRISM model checker [Kwiatkowska et al., 2002].

Our implementation of Algorithm 2 is referred to as **RCA**. RiskNeutralPolicy is implemented via value iteration. Computations of $E[\mathcal{R}_{i,\pi_i}]$ and $E[\mathcal{C}_{i,\pi_i}]$ are computed with 1000 Monte Carlo trials. CalcRisk is done by first calculating the VaR with 1000 Monte Carlo trials. Then the CVaR and risk contributions are calculated via rejection sampling: first 1000 Monte Carlo trials are run continuously until there are 1000 satisfying trials that exceed the VaR, then these 1000 trials are used to estimate the CVaR and risk contributions. Stepsize γ is set proportionally to C. The Joint Reward and Joint VaR displayed in the results graphs are also calculated with 1000 Monte Carlo trials, and the Joint CVaR is calculated in the same way as the CalcRisk method, though over the joint state space. All methods were implemented in Python. All experiments were conducted on an AWS R5a.large EC2 instance, with 2 CPUs and 16GB of memory.

5.3.2 The Maze Domain

Domain Description.

The Maze Domain is as described in Chapter 3.3.1, originally from Wu and Durfee [2010]. As before, the global time horizon h is 2 times the width of the grid. We set the confidence interval to $\delta = 0.05$ in all experiments. The limit on the joint CVaR is $L = \frac{hn}{4}$, where h is the global time horizon and n is the number of agents in the system. All methods timeout at 5000 seconds. This domain contains many possible configurations of start and end locations; as such each data point represents the average results from 50 possible configurations. Distributional information on experiments can be found in the Appendix.

Results.

As expected, iRMDP scales poorly with respect to planning time, as seen in both Figure 5.1, which varies the size of the single-agent state space, and Figure 5.2, which varies the number of agents. Both these variables have the effect of increasing the joint multi-agent state space. In both cases, iRMDP is unable to solve instances larger than 2 agents and a 5 by 5 gridsize. Note that for Figure 5.2 this is only a

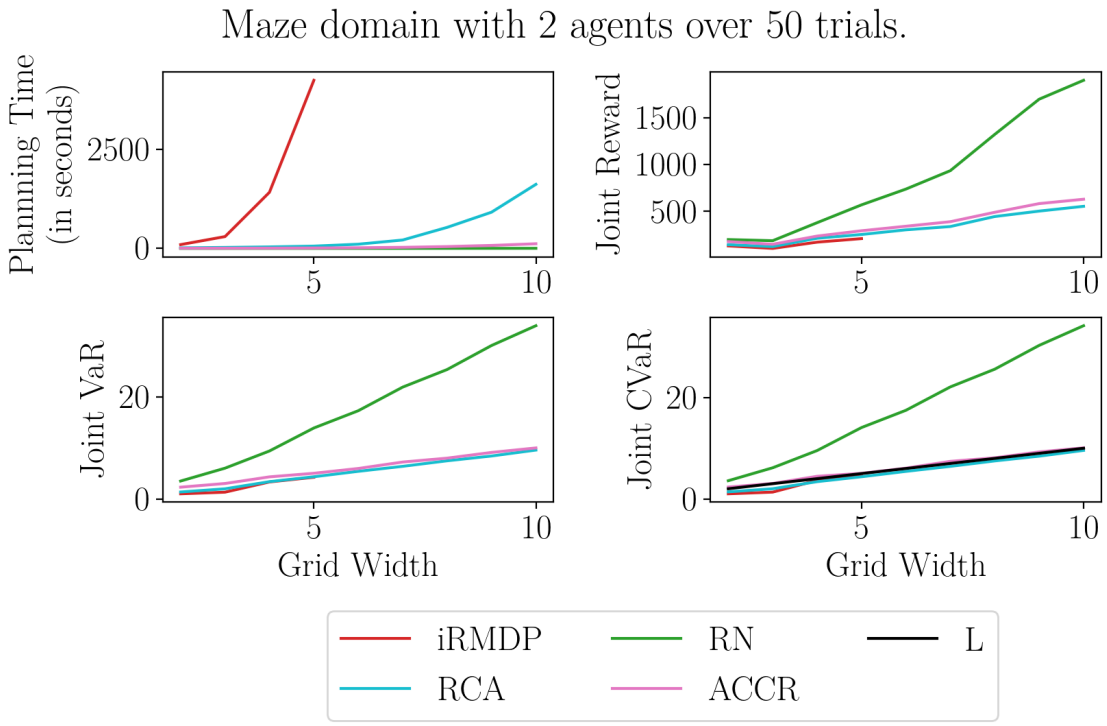


Figure 5.1: An analysis of RCA and iRMDP on increasing large instances of the Maze domain for 2 agents. Each data point corresponds 50 trials. See the Appendix for this chart with error bounds.

single point on the graph. Because the time horizon is set to $h = 10$ for this gridsize, this is equivalent to 500 states. The solution value from iRMDP is also slightly less than RCA. This is due to the convergence settings for iRMDP, 0.01 for both the middle and outer loop, whereas RCA's RCConstrainedPolicy has convergence settings of 0.001 for both the middle and outer loop. While a finer condition would improve the solution value, it will also increase the planning time, and thus we choose the current condition to trade off between the two. Note that despite the approximation RCA makes, RCA still achieves close-to-optimal performance in problem instances where the optimal can be evaluated. In both Figures, RN outperforms both iRMDP and RCA in terms of time and joint reward, but only because it ignores the CVaR constraint.

The CVaR is constrained by L as expected for iRMDP and RCA methods, in both Figures 5.1 and 5.2. ACCR appears to perform best in Figures 5.1 and 5.2; it has a low computational time and appears to also consistently constrain CVaR at or

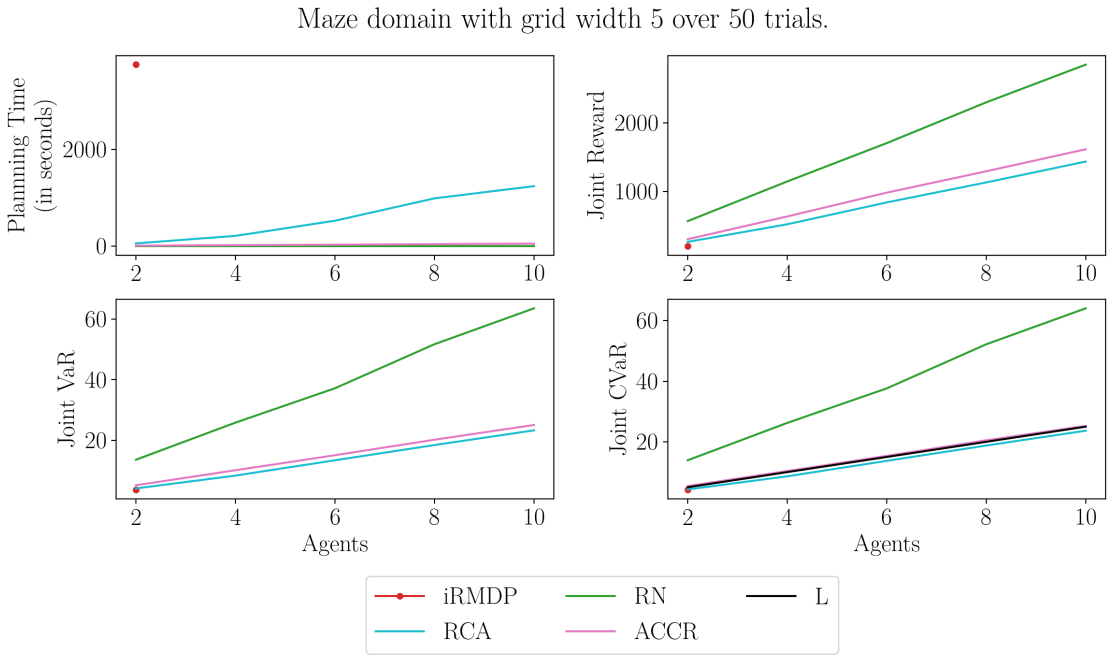


Figure 5.2: An analysis of RCA and iRMDP on increasing numbers of agents in the Maze domain of gridsize 5 by 5. Each data point corresponds 50 trials. See the Appendix for this chart with error bounds.

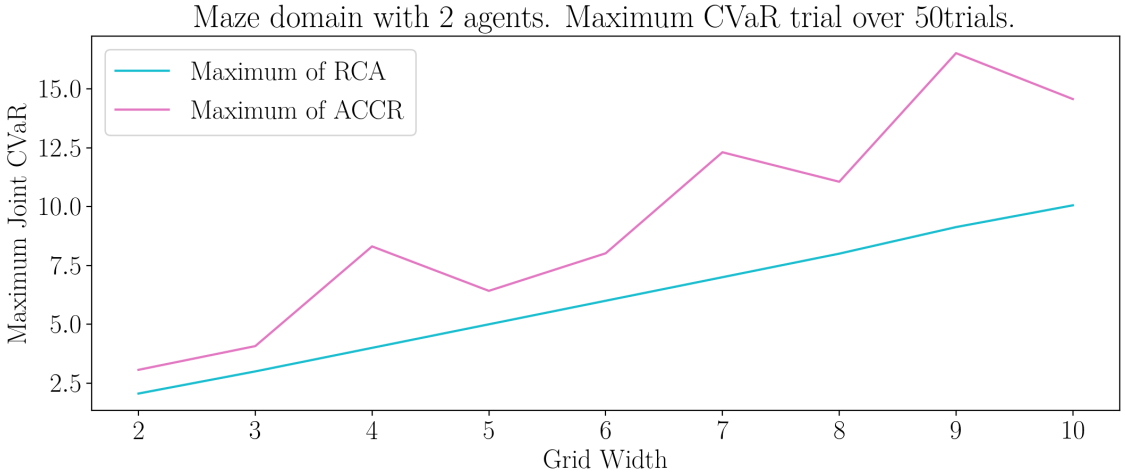


Figure 5.3: The maximum CVaR trial on the Maze domain for 2 agents.

below the limit L. But data points in these two charts represent the average over a 50 trials, and when considering ACCR distribution over at each data point ACCR is not guaranteed to satisfy the CVaR constraint. In Figures 5.3 and 5.4, we see the CVaR of the maximum ACCR trial always exceeds the CVaR bound. These results are as is expected because ACCR is planning for a chance constraint, not a risk constraint.

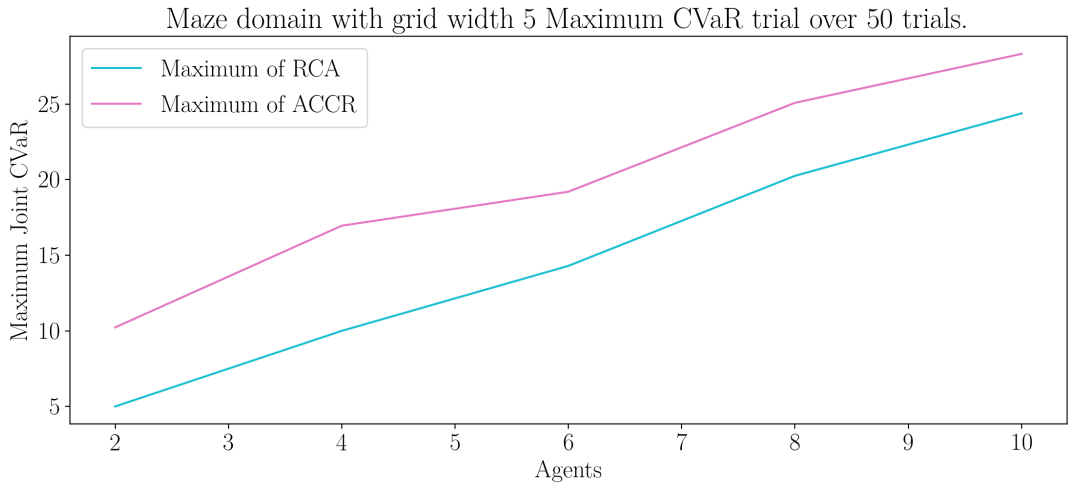


Figure 5.4: The maximum CVaR trial on the Maze domain for grid width 5.

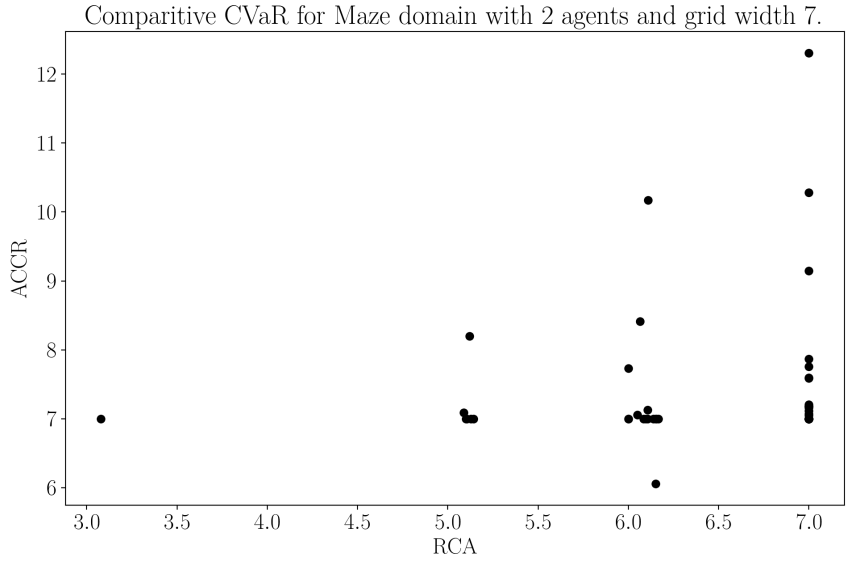


Figure 5.5: ACCR vs RCA: CVaR for 2 agents on grid width 7. Each dot represents the CVaR for one trial. $L = 7$.

We can also examine the comparative CVaR for any given Maze setup. In Figure 5.5, we examine a single setup of Maze with 2 agents and grid width 7, where each dot represents a single trial. In this setup, $L = 7$. Note that in every trial, the CVaR for RCA is less than or equal to 7, but the CVaR for ACCR frequently exceeds 7. The CVaR for RCA are clustered around integers because of the RCA step size γ.

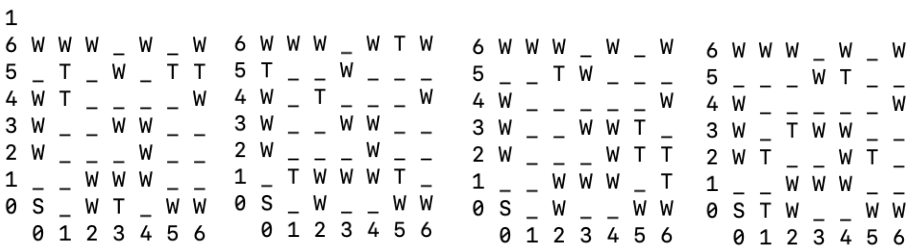


Figure 5.6: Four agent MDP setups on the Maze domain. S corresponds to the start location, T corresponds to task spaces, and W corresponds to walls.

A Closer Look at Policies

To better investigate the policies returned by each method, we focus on a specific instance of Maze, with grid size 7 and 4 agents with the setups shown in Figure 5.6. In this setting, $\delta = 0.05$ and $L = 14$.

In Figure 5.7, 5.8, and 5.9, we consider the individual agent reward distributions $\mathcal{R}_{i,\pi_i}$ and joint reward distribution $\mathcal{R}_\pi$, individual cost distributions $\mathcal{C}_{i,\pi_i}$ and joint cost distributions $\mathcal{C}_\pi$ for RCA, ACCR, RN respectively. In Figure 5.7, we see that for RCA, Agent 3 and 4 have been allocated resources, while Agent 1 and 2 have been allocated no resources. This occurs because Agent 3 and 4 had better risk-to-reward ratios, and thus the risk contribution of Agent 1 and 2 were iteratively lowered until the risk constraint was met. In Figure 5.8, we see that for ACCRR, Agent 1 has been allocated 7 resources with a $\delta = 0.05$ probability of exceeding those resources. ACCR is agnostic to how many resources Agent 1 exceeds by, but in practice this is by 13 resources. No other agents ever exceed their resources, and all agents almost always use their full allocated number of resources. This results in a multi-model joint cost distribution where there is close to a $\delta = 0.05$ probability that the joint cost distribution will be at least 20, meaning that the CVaR for $\delta = 0.05$ well exceeds the resource limit of $L = 14$. In Figure 5.9, we see that for RN, agents' resource usage is unconstrained, and thus agents can use as many resources as they need. This results in agents using safe actions at every timestep. Because safe actions are still stochastic, the tasks they are able to achieve still changes in each run, as does the time to takes to achieve those tasks. But, in

most cases agents are able to achieve their maximum reward task, and it takes almost the maximum possible resource use to do so.

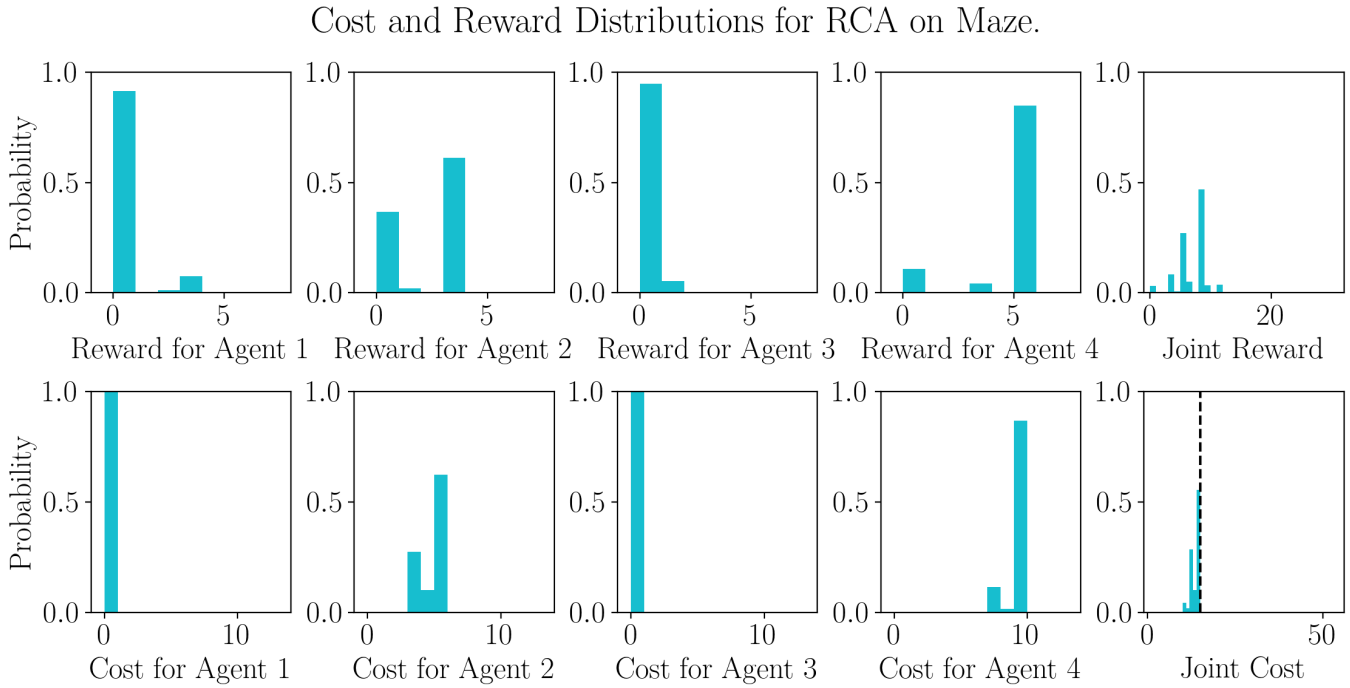


Figure 5.7: From top to bottom and left to right: individual agent reward distributions and joint reward distribution for RCA; individual cost distributions and joint cost distributions for RCA. Agent setups are described in Figure 5.6.

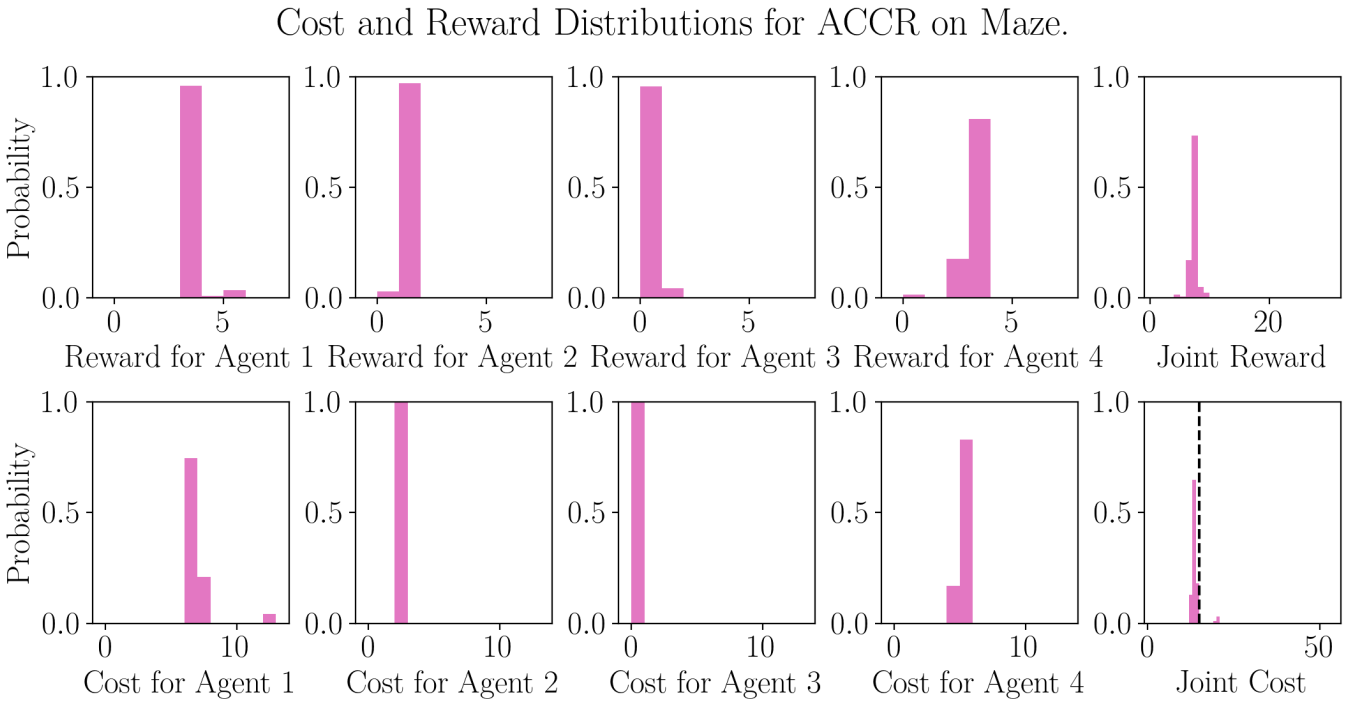


Figure 5.8: From top to bottom and left to right: individual agent reward distributions and joint reward distribution for ACCR; individual cost distributions and joint cost distributions for ACCR. Agent setups are described in Figure 5.6.

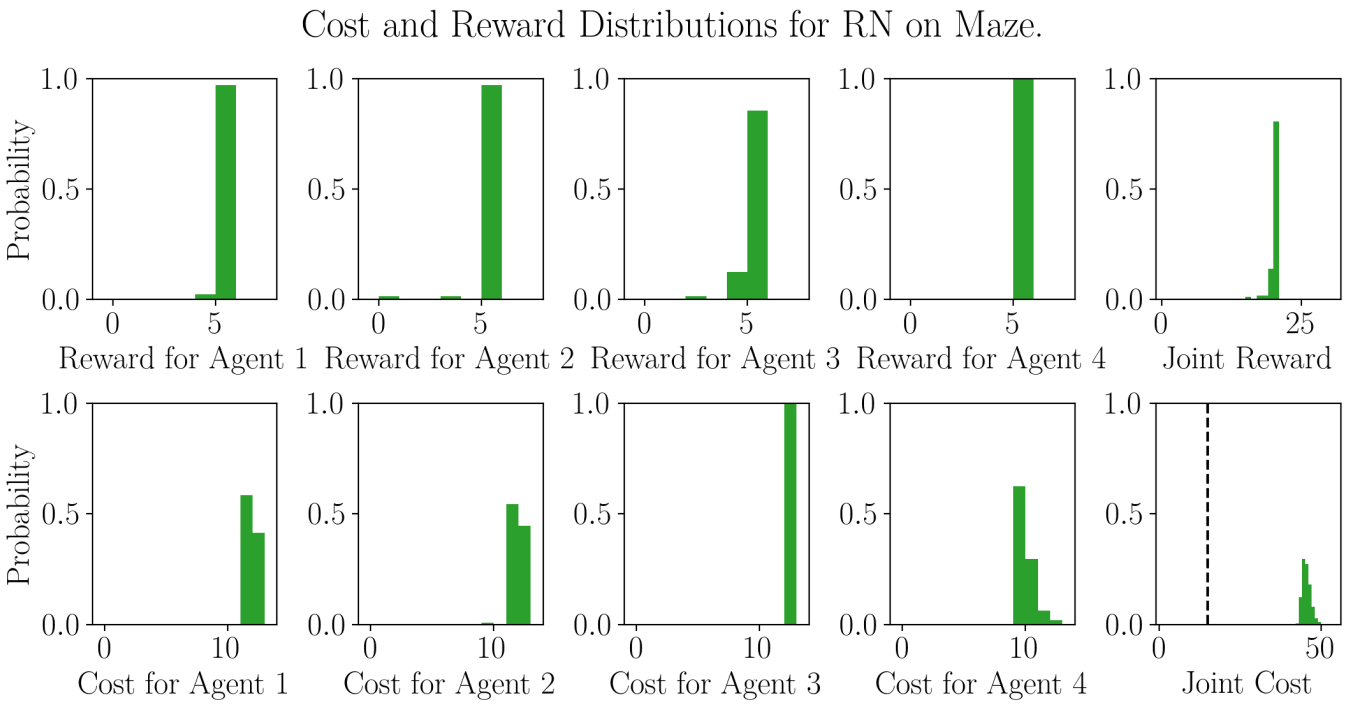


Figure 5.9: From top to bottom and left to right: individual agent reward distributions and joint reward distribution for RN; individual cost distributions and joint cost distributions for RN. Agent setups are described in Figure 5.6.

5.3.3 The Advertising Domain

Domain Description.

The Advertising domain is as described in Chapter 3.3.1 and Example 5, originally from [Boutilier and Lu, 2016], with 1000 agents, 15 states, and 5 actions per state. Unlike in Chapter 4 , we slightly modify the domain by setting a time horizon of $h = 30$ to make the more difficult risk-constrained problem feasible to solve. This MDP contains only one configuration (as in [Boutilier and Lu, 2016]), and as such results are over a single execution of RCA. As before, in all experiments, $\delta = 0.05$. In Figure 5.10 the joint CVaR is limited by $L = 10n$. In Figure 5.11, L is varied along the x-axis.

Results.

As in the Maze domain, in Figure 5.10 the planning time for RCA increases with the number of agents (and the increased CVaR requirements). The CVaR is successfully constrained through all numbers of agents in Figure 5.10. In Figure 5.10, we see how the results of RCA vary for a consistent number of agents as the constraint on CVaR (L) changes. Because RCA starts with a risk-neutral policy and then iteratively decreases the joint CVaR, RCA plans faster as the budget increases. As expected, the solution value increases as the budget increases because agents are allowed more flexibility in their action choices. The CVaR of ACCR is successfully constrained through all numbers of budgets in Figure 5.11. In both Figures, RN once again outperforms RCA in terms of time and joint reward, but again only does so because it ignores the CVaR constraint. In this domain ACCR performs best in both Figures 5.10 and 5.11; it has a low computational time and consistently constrains CVaR at or below the limit L. ACCR is planning for a chance-chance constraint and not a risk-constraint. But if we consider ACCR's empirical performance in Chapter 4 on this domain in Figures 4.9 and 4.10, ACCR consistently under-utilised its resource budget, meaning that the actual probability of resource violations was significantly less than the required 0.05. Given this signifiant under-utilisation of the chance-constraint budget, it is unsurprising that it also satisfies a risk-constraint budget.

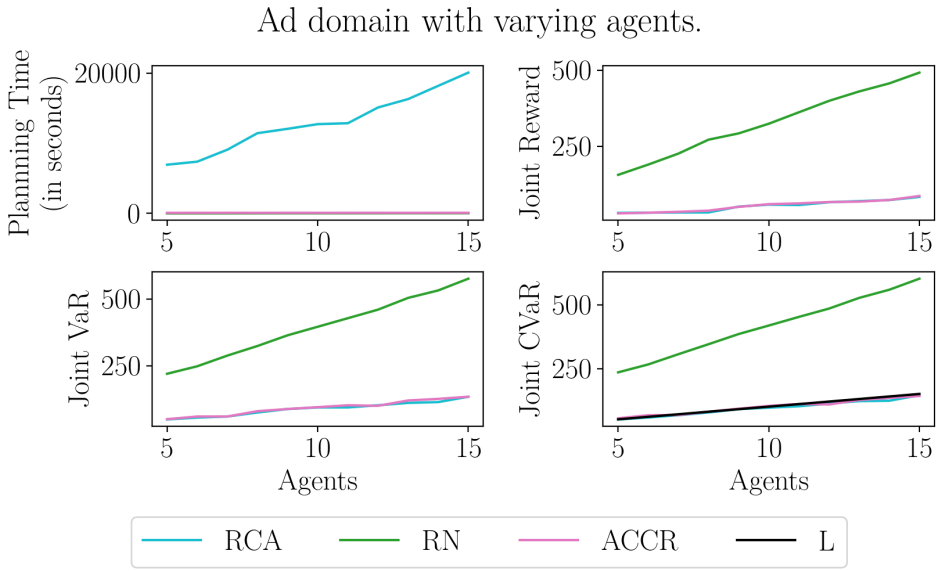


Figure 5.10: Algorithm performance on the Advertising domain with increasing numbers of agents.

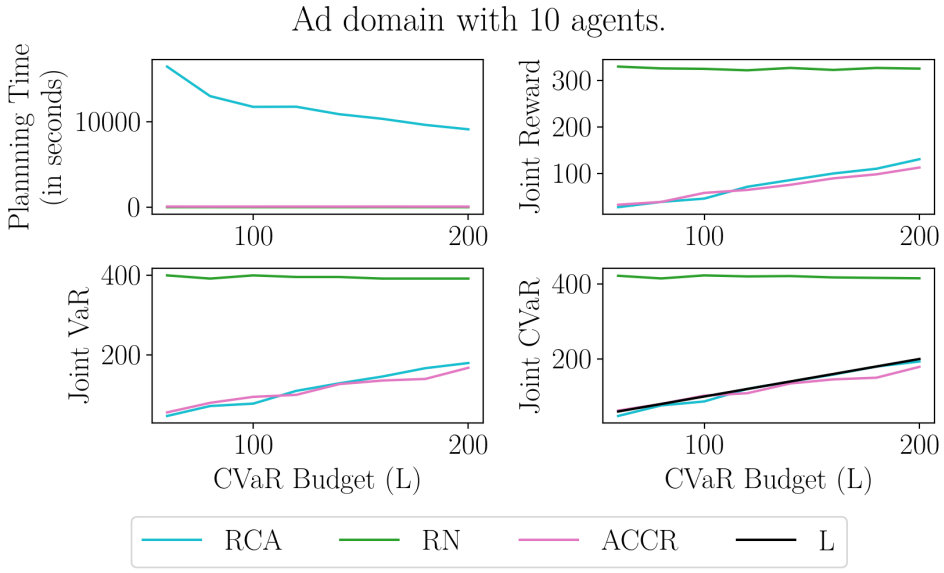


Figure 5.11: Algorithm performance on the Advertising domain with increasing budgets.

5.4 Discussion

5.4.1 Summary

Constrained planning problems in multi-agent systems under uncertainty require unique solutions to handle the state space explosion that arises from MMDPs. We defined the Risk-Constrained MMDP problem, i.e., the problem of optimising for expected reward while constraining the joint CVaR of a shared resource in a cooperative setting (Section 5.2.1). This constraint can be interpreted as limiting the expected value of a shared resource in the worst cases. To do this, we introduced a concept from finance called risk contribution, which allows us to identify agents who contribute proportionally more risk to reward. We then update the worse performing agent's policy to iteratively lower their risk contribution, and thus the joint CVaR (Section 5.2.3). With this method, we avoid the state and action space explosion of solving a joint model by instead iteratively updating only one agent's policy (Section 5.2.4). We evaluated RCA against a single-agent risk-constrained solver iRMDP on two benchmarks, the Maze domain from Wu and Durfee [2010] and an advertising domain from Boutilier [1996]. We demonstrate that not only can RCA successfully solve the RCMMDP problem, but also it significantly outperforms iRMDP in terms of planning time (Section 5.3).

5.4.2 Limitations and Future Work

In Chapter 4 we discussed the chance-constrained resource allocation problem, and in this chapter we moved to a richer understanding of uncertainty with the risk-constrained resource allocation problem. One limitation to RCA is the added computation time in evaluating this richer notion of uncertainty. While RCA is much less computationally intensive than solving the joint risk-constrained model, it is still much more computationally intensive than a risk agnostic method or chance-constrained method. There are two main contributors to the time RCA takes, the single-agent risk contribution algorithm and the number of repetitive calls to the single-agent risk contribution. While better single-agent methods would

significantly reduce the computation time, computational techniques like warm-starting RCConstrainedPolicy with λ^0 and $\delta^{w,0}$ may also reduce the computation time. Additionally, the number of calls to RCConstrainedPolicy may be lowered by increasing risk contribution reduction γ at each call, but this will come at the cost of solution value.

Another limitation of RCA comes from the approximation made by considering only the previous VaR and previous risk contribution of the other agents when replanning to reduced the selected agent's risk constraint. One way to avoid this would be to induce Discrete Time Markov Chains (DTMCs) from the other agents' policies and MDPs, and then plan on a joint model that consists of the single agent's MDP and the other agents' induced DTMCs. This would result in a single-agent policy that would be dependent on the other agents' states. So while it would avoid an approximation, it would result in an increased computation time as a result of planning over the joint state space. As a result this approach would scale poorly as the number of agents increases. It would also require either a centralised execution or an execution where all agents have full observability over the other agent's state spaces. While this is a promising extension to our work, this is a separate stetting from the weakly-coupled MMDP with Constraints we have considered in this thesis.

Finally we discuss the possibly of solving the non-cooperative risk-constrained multi-agent planning problem. Extending RCA to a non-cooperative context is a non-trivial problem. The risk-constrained multi-agent resource allocation is ill-suited to an auction based mechanism because agents' risk contributions are heavily dependent on the other agents' cost distribution random variables ($\mathcal{C}_{i,\pi_i}$). In Section 5.2.3 we discuss how to estimate risk contributions with summary statistics from the other agents' $\mathcal{C}_{i,\pi_i}$, instead of exactly calculating an agent's risk contribution with the other agents' full distribution. But even with this approximation, there is still an implicit dependence on the other agents' policies π_j through the var_{u-1} and $rc_{u-1,i}$ summary statistics. As such it is impossible for an agent to include their risk contribution in a bid in the same way the probability of exceeding is included in ACCR in Chapter 4. This motivated our search for a centralised, iterative solution like RCA.

A final important area for future work is considering other coherent risk measures, which is a class of notions of risk with desirable properties of which VaR and CVaR are members.

6

Allocating Space: A Non-Cooperative Approach to MAPF

6.1 Introduction

In this chapter, we move away from uncertain environments with simple resource types and instead focus on deterministic environments with multiple resources types. In particular, we consider the area of Multi-Agent Pathfinding (MAPF), where agents share physical space and need to avoid conflicts, i.e., being in the same space at the same time. Classical cooperative multi-agent pathfinding focuses on preventing these types of conflicts through the use of a centralised mechanism. Though these robots have collision avoidance for non-stationary objects (like humans), relying on collision avoidance for robot-robot interaction places an unnecessary burden on low-level controllers and can cause significant delays or even collisions [Street et al., 2021].

Example 6. *Consider a superstore, with aisles of groceries, clothing, and electronics. The store does not have the technology to build a fleet of robots, so they contract out individual tasks. One company might be hired to deploy a robot to move around the store examining shelves to track inventory. Another company might be hired to deploy cleaning robots. These robots have misaligned incentives; while they both have to interact in the space, they only care about their own contract with the store, and*

Table 6.1: A description of Chapter 6's relevance to the three main challenges of multi-agent decision making.

	Chapter 4	Chapter 5	Chapter 6
Uncertain Domain?	Yes	Yes	No
Rich Uncertainty?	No	Yes	No
Multiple Resource Types?	No	No	Yes
Non-Cooperative?	Yes	No	Yes

so their only goal is to complete their own task. They may be penalised for failing to complete their tasks either with a built-in monetary penalty or the eventual loss of their contract.

Systems such as the one in Example 6 *non-cooperative*. In non-cooperative settings as defined in Section 2.2.2, agents are both rational and strategic. Because agents are rational, they are interested in optimising only their own social welfare function. In a MAPF context, this means agents are concerned with the length of their own path. Additionally, because agents are strategic, they are unwilling to blindly follow a centralised path planner and may manipulate the system in order to benefit themselves. We consider this non-cooperative variant of the MAPF problem.

While multi-robot pathfinding problems [Stern et al., 2019] are well-studied (see Section 6.2.1) very little research to date has considered the non-cooperative case. One notable exception is the work of Amir et al. [2015], which introduced a mapping between MAPF and combinatorial VCG auctions. They then propose the use of *iBundle* [Parkes, 1999] to find solutions to the MAPF problem. iBundle, as described in Section 2.2, is an iterative combinatorial auction algorithm which, under certain conditions, provides equivalent solutions to VCG through decentralised conflict resolution. However, iBundle has a number of limitations. First, iBundle relies on agents continuing to submit and value more paths if the price of their high value paths become too large due to conflicts. As a result, iBundle requires agents to have access to an ordered list of all possible single solutions ranked by descending value. Depending on how agents assign value to different solutions, this could require agents to evaluate all possible solutions, or in the case where value functions are tied to shortest paths, this requires the agents to have the

ability to access the next shortest path at any given time. Second, the bidding strategy that agents are expected to take will slow down iBundle in the MAPF setting. An agent's myopic best response strategy to the iBundle auction, i.e., the bidding strategy which makes the most sense for them to execute, requires them to submit all paths of equal length at the same time. Because there can be many paths with the same length, agents are incentivised to submit a large set of bids at once, and this detracts from the iterative advantage of iBundle and can lead to a high computation time. Finally, because iBundle finds the optimal allocation, and because the winner determination sub problem of VCG is NP-complete [Dobzinski and Nisan, 2007], iBundle is also NP-complete. All three of these issues taken together result in a prohibitively large worst-case computation time.

In this chapter, we build on Amir et al. [2015] and tackle the limitations listed above. We do this by designing a mechanism for non-cooperative path planning that exploits centralised conflict resolution via the *privileged knowledge* it obtains from the initial agent bids (Section 6.3). This heuristic mechanism is designed to optimise for social welfare while incentivising agents to participate. Our mechanism is tailored for the MAPF problem, allowing agents to submit a smaller number of paths to the auction, which helps us overcome the limitations of iBundle (Section 6.3.1). The auctioneer takes on the burden of removing conflicts, and uses a method specific to MAPF to do so (Section 6.3.2). We also consider the problem of single-agent bid generation, and propose an adaptation of the dissimilar path search algorithm from Jeong and Kim [2011] (Section 6.4). Our adaptation modifies the similarity metric to better suit a MAPF context by considering both spatial and temporal similarity. Because agents submit a smaller number of bids, using a dissimilar path algorithm allows agents to provide the auctioneer with a wider range of possibilities, which makes a conflict-free solution in the initial stage of the auction more likely. We analyse our mechanism and single-agent bid generation on two synthetic domains: a model of a warehouse and maps from *Dragon Age: Origins* [Stern et al., 2019]. Because our mechanism relaxes guarantees and optimality

to improve computation time, it outperforms iBundle in almost all cases when under a computation time constraint (Section 6.6).

This chapter describes the work published in Gautier et al. [2022] in the Proceedings of AAMAS 2022. Section 6.4.2 was completed in collaboration with Alex Stephens, and is included to provide important context for the results in Section 6.6.

6.2 Multi-Agent Pathfinding

First, we formally define the MAPF problem and discuss the current literature. MAPF is a generalisation of the classic Shortest Path Problem to n agents. Agents act on a graph $\mathcal{G} = (V, E)$, where V is a set of vertices and $E \subseteq V \times V$ is a set of edges. Each vertex $x \in V$ represents coordinates in an environment, i.e., $x \in \mathbb{R}^2$.

Each agent i starts at their own unique start vertex r_i, and at each discrete timestep agents can wait in their current vertex or travel to another vertex that is connected to their current position by some edge in E. Each agent has a unique goal vertex g_i. All definitions in this section originate from [Stern et al., 2019].

We denote the set of all finite sequences of elements of V as V^*, and define a *valid* single-agent path on $\mathcal{G}$:

Definition 32 (Valid Path)**.** *A valid path for an agent i is a sequence of vertices $w_i = x_{i,1}x_{i,2}\dots x_{i,\ell_i} \in V^*$ where:*

- $(x_{i,d}, x_{i,d+1}) \in E$ *for all* $d \in \{1, \dots, \ell_i - 1\}$,
- $x_{i,1} = r_i$, *and*
- $x_{i,\ell_i} = g_i$.

The goal of the planner is to find *optimal*, valid, *conflict-free* paths for every agent.

First, we address the question of optimality in MAPF solutions. There are a few classic MAPF objectives, but the two most common are *flow time* and *makespan*.

Definition 33 (Flow time)**.** *Given a set of valid single-agent paths, represented by $\{w_i\}_{i\in[n]}$, the flow time is defined by $\sum_{i\in[n]} \ell_i$.*

Flow time is the sum of the path lengths, and as a result, is often also referred to as the sum of cost.

Definition 34 (Makespan)**.** *Given a set of valid single-agent paths, represented by* $\{w_i\}_{i\in[n]}$*, the makespan is defined by* $\max_{i\in[n]} \ell_i$*.*

Makespan represents the timestep at which the last agent will reach their goal. For the remainder of this thesis we will focus on flow time as it is often the first optimisation function considered in any new algorithm [Sharon et al., 2015, Standley, 2010], and mirrors the objective of social welfare that we considered in Chapters 4 and Chapter 5. We discuss extensions to makespan in Section 6.7.2.

Next, we consider what it means for a solution to be conflict-free. A *conflict* occurs between two paths when those two agents interact on the graph at a given time. Examples of conflicts include: *vertex conflicts*, *edge conflicts*, and *swapping conflicts*. These conflicts are defined with respect to two agent paths. Say there are two agents and agent i takes valid path w_i and agent j takes valid path w_j. Then:

Definition 35 (Vertex Conflict)**.** *A vertex conflict occurs if there exists some* $d \in \{1, \ldots, \ell_i - 1\}$ *such that* $x_{i,d} = x_{j,d}$*.*

A vertex conflict occurs when two agents occupy the same vertex at the same time.

Definition 36 (Edge Conflict)**.** *An edge conflict occurs if there exists some* $d \in \{1, \ldots, \ell_i - 1\}$ *such that* $x_{i,d} = x_{j,d}$ *and* $x_{i,d+1} = x_{j,d+1}$*.*

An edge conflict occurs when two agents traverse on the same edge in the same direction at the same time.

Definition 37 (Swapping Conflict)**.** *A swapping conflict occurs if there exists some* $d \in \{1, \ldots, \ell_i - 1\}$ *such that* $x_{i,d} = x_{j,d+1}$ *and* $x_{i,d+1} = x_{j,d}$*.*

A swapping conflict occurs when two agents traverse on the same edge in opposite directions at the same time, i.e., if there exists some $d \in \{1, \ldots, \ell_i - 1\}$ such that

$x_{i,d} = x_{j,d+1}$ and $x_{i,d+1} = x_{j,d}$. Much of MAPF community refers to a swapping conflict as an edge conflict, but here we follow the convention of [Stern et al., 2019].

What interaction results in a conflict can depend on the domain. For the remainder of this thesis we will focus on vertex conflicts. The set of edge conflicts is a subset of the set of vertex conflicts, and thus when planning to avoid vertex conflicts one also avoids edge conflicts. While we do not consider swapping conflicts directly, our work in Chapter 6 can be generalised to swapping conflicts as discussed in Section 6.7.2.

Thus, we define a *conflict-free* solution $\{w_i\}_{i\in[n]}$, as a set of paths such that there does not exist any pair of paths that contain a vertex conflict.

We have presented here the most common definitions of optimality and conflict in the literature, for additional variants of MAPF, see Stern et al. [2019].

6.2.1 Existing Work on MAPF

Cooperative MAPF is widely studied, and several algorithms have been proposed for it. The most efficient optimal algorithm is Conflict Based Search (CBS) which allows agents to start with their shortest path (agnostic to other agents) and then builds a tree of conflicts and possible resolutions as agents iteratively replan based on conflicts [Sharon et al., 2015]. Quicker, but sub-optimal solvers exist. Prioritised planning with reservation tables allows one agent to plan at a time, and then removes their space-time occupancy from the map before the next agent can replan. [Silver, 2005] introduces Hierarchical cooperative A*, which is an early, quick but suboptimal solution to the problem based on A* search.

The MAPF problem has been extended in a variety of ways, for example to consider uncertainty [Wagner and Choset, 2017, Shahar et al., 2021], congestion [Street et al., 2021], and kinematic constraints [Hönig et al., 2016, Walker et al., 2017]. For more details on cooperative MAPF, see Stern et al. [2019].

Non-cooperative MAPF was introduced by Amir et al. [2015]. They introduce an equivalence between the non-cooperative MAPF problem and combinatorial VCG auctions, which is discussed in depth in the below. This equivalence allows

Table 6.2: A high level overview of the parallel between MAPF and combinatorial auctions (CA) adapted from Amir et al. [2015].

MAPF	CA
Vertex-time pair	Item
Path (set of items)	Bundle
Path cost	Valuation function
Minimal sum of path costs	Maximal social welfare

them to solve MAPF problems with a combinatorial VCG auction. However, the combinatorial nature of the problem can lead to inefficiencies, even when using an auction like Parkes [1999]. Chandra et al. [2022] introduce an online non-cooperative MAPF planner with a one step lookahead that breaks local conflicts with an auction mechanism.

6.2.2 MAPF as a Combinatorial Auction

We now describe how a MAPF problem can be solved using a combinatorial auction, as introduced by Amir et al. [2015]. The overall idea is to consider *paths as bundles*. We start by noting that a path $w = x_1 x_2 \dots x_\ell$ can be interpreted as a bundle of vertex-timestep pairs, i.e., $\{(x_1, 1), (x_2, 2), \dots, (x_\ell, l)\} \subseteq V \times \mathbb{N}$. Thus, for notational convenience, for the remainder of the chapter we will denote paths interchangeably as either bundles or sequences, i.e., we will denote a l-length path for agent i as $A_i = \{(x_1, 1), (x_2, 2), \dots, (x_\ell, l)\} \subseteq V \times \mathbb{N}$ or as $w_i = x_1 x_2 \dots x_\ell \in V^*$. Agents value these bundles based on how effective the corresponding path is in achieving their goal and what cost it takes them to traverse that path. When optimising for flow time, path cost is equivalent to path length if agents are moving at a constant speed. We define the best MAPF solution as one in which the sum of agents' costs is as low as possible. The minimum cost allocation of a MAPF problem is equivalent to the maximum value allocation of a combinatorial auction. Thus, maximising the sum of all values over all agents, known as the *social welfare*, results in a solution to the MAPF problem in which as many agents reach their goals with as little cost as possible. A high-level overview of the parallel between MAPF and combinatorial auctions can be seen in Table 6.2. Determining feasible allocations

via a combinatorial auction ensures that agents do not enter into conflicts. If two agents' bundles S_i and S_j have no intersection, then they will never be at the same vertex at the same time.

Because each vertex-time pair is a unique, non-fungible resource, the allocation algorithm must consider both the differing resource type and the coupling between them. If an agent is interested in taking a path $S_i = \{(x_1, 1), (x_2, 2), (x_3, 3), \ldots, (x_\ell, l)\}$ the vertex-time pair $(x_3, 3)$ is only worth something if they are also allocated $(x_2, 2)$ right before it. This coupling means that agents need to express bids over entire bundles of resources, making the problem combinatorial and complex.

Unfortunately, determining the welfare-maximising outcome for VCG is NP-complete [Dobzinski and Nisan, 2007], so solving this problem optimally is time consuming. Additionally, using combinatorial VCG auctions for MAPF requires agents to give a value for every possible path. Because the number of paths is on the order of $O(2^{|V|\times h})$ for $|V|$ vertices and h timesteps, agents need to determine their bid value for $O(2^{|V|\times h})$. This puts an unrealistic computational burden on the agents, who may have more complex valuation functions than just path length. It also increases the size of the Combinatorial ILP described in Section 2.2.1. The iterative algorithm iBundle presents an alternative to an ILP and allows agents to value fewer paths, but requires the ability to list paths in value order. Another limitation comes from the strategy that agents are expected to use in the iBundle auction. In an iterative auction like iBundle, agents' strategies determine what bids to make at each iteration of the auction. Parkes [1999] show that the *myopic best response strategy* for iBundle (i.e., the strategy agents use when considering only the outcome of the current iteration) requires agents to submit all bids with equivalent value at the same time. But the nature of the MAPF problem means that many paths will have the same value, and thus in practice agents bidding in iBundle will still be required to submit a large number of bids at each iteration of the iBundle algorithm.

6.3 Privileged Knowledge Auction

We now introduce a new mechanism called the privileged knowledge auction (PKA) which removes the problem of enumerating every bid of the same value at the same time, by allowing agents to place a fixed number of bids, say m. Our aim is to create a mechanism to solve the non-cooperative MAPF problem. To be a MAPF solution, a set of paths must avoid conflicts and allow agents to reach their goals. To be a non-cooperative solution, agents must have autonomy over their own paths and the system should be difficult to manipulate by strategic agents. Figure 6.1 provides an overview of our proposed mechanism. Agents submit bids and the auctioneer carries out a modified VCG auction (Section 6.3.1). Then the auctioneer uses knowledge from the sealed-bid VCG auction, which we refer to as *privileged knowledge*, to find a solution outside of the submitted proposals if necessary (Section 6.3.2). Finally, the auctioneer gives agents autonomy over what bids are chosen through a descending auction (Section 6.3.3).

6.3.1 Initial Bidding and VCG Auctioning

Agents propose timed bundles in the form $\{(x_1, 1), (x_2, 2), ...(x_\ell, \ell)\}$. In the first step of the mechanism, each agent proposes a fixed number of paths to the auctioneer. In practice, this number will typically be small to cope with the scalability issues of VCG and is chosen as a design parameter. The auctioneer then searches for a potential feasible allocation using the same winner determination and pricing as combinatorial VCG. At this stage of the auction, agents can only be allocated bundles which they themselves proposed. As a result, all allocations will correspond to actual paths. If no solution is found in which every agent is allocated a path, we move into the second part of our protocol.

6.3.2 Deconflicting

In the second part of PKA, the auctioneer uses the privileged knowledge gained in the bidding stage to find a better solution. If the auctioneer cannot find a solution given the proposed paths supplied by the agents, they must instead generate their

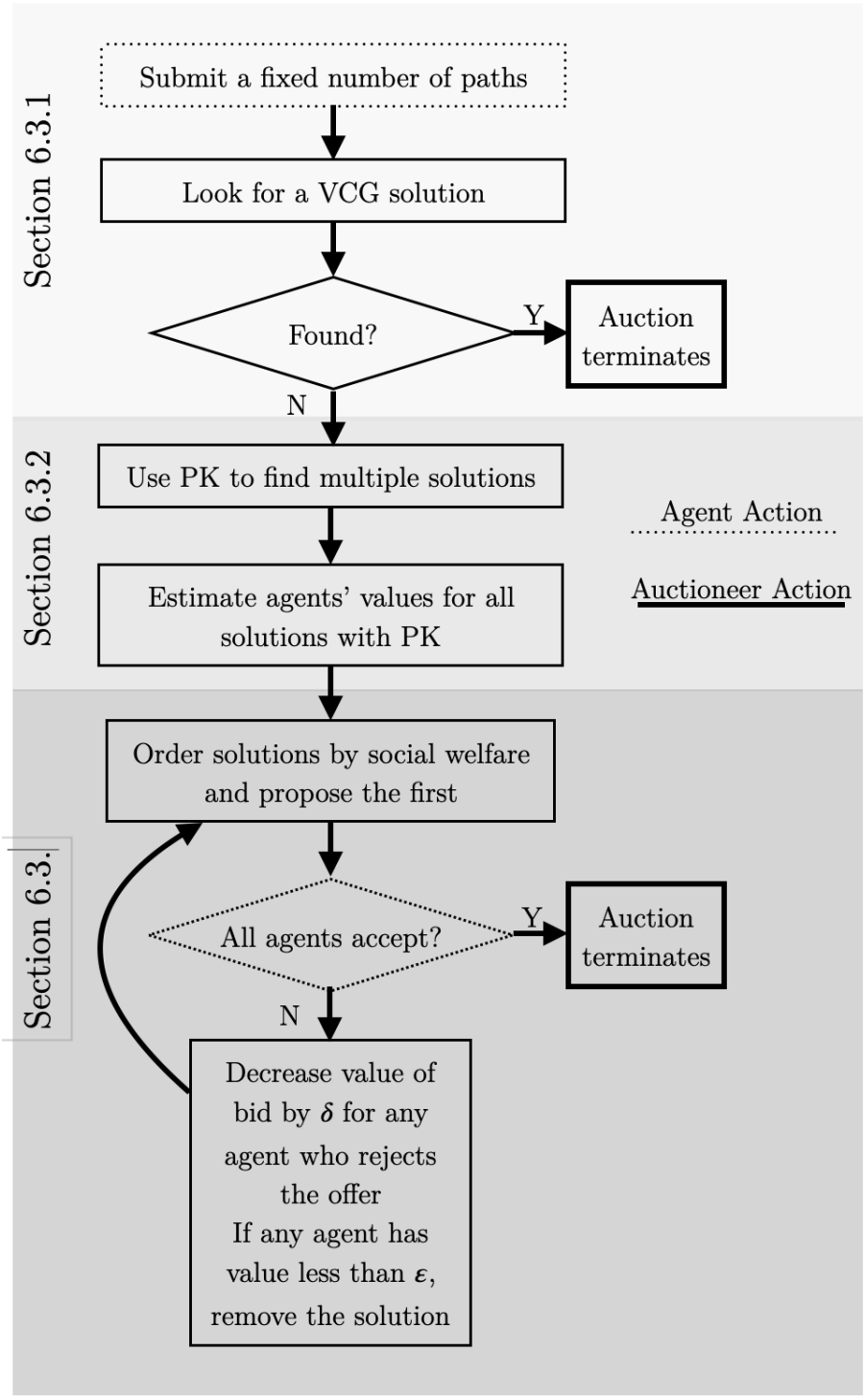


Figure 6.1: A high level overview of our PKA protocol.

own, conflict-free solution. In order to preserve the agents' autonomy, the auctioneer generates multiple alternate conflict-free solutions from which the agents can choose. Before we discuss the incentive-aware process that the auctioneer uses to choose

between possible solutions, we address how those solutions can be generated. Our mechanism is agnostic to the method used to find such solutions, the suitability of each possible method being tied with the particular environment. In situations where the auctioneer believes value is closely tied to the shortest path to the goal, classical cooperative MAPF algorithms are a good choice. Any MAPF algorithm (or set of MAPF algorithms) which can be designed to return a valid set of solutions is suitable. Many algorithms are suited to this purpose, and the choice of algorithm is an input of PKA. In our experiments, we used hierarchical cooperative A* (HCA*) [Silver, 2005]. HCA* is a quick, suboptimal, solution algorithm where agents plan their paths sequentially in a graph extended to include both space and time. After each agent plans their path, the locations they pass through on the graph are removed from the graph at that particular time, and the next agent is allowed to plan their path. Agents are allowed to plan their paths in different orders to produce different solutions. We leave comparing different solution methods to future work. Ideally, we desire at least two solutions from the MAPF solver but if there is only one possible solution, the auctioneer will proceed with only that one solution.

Assuming that the auctioneer has a set of alternate solutions $\mathcal{AS} = \{\{A_j^1\}, ..., \{A_j^f\}\}$, it is necessary to order them from most desirable to least desirable. The goal of the mechanism is to elicit truthful values, so while we are unable to extrapolate the exact social welfare of these solutions with the partial value information obtained in the first stage of the mechanism, we can develop a heuristic, and then use that to elicit true preferences by asking the agents for more information (Section 6.3.3). To approximate the social welfare of a specific alternate solution $\{A_j\}$, we need to approximate the value of each path $A_i \in \{A_j\}$ allocated to each agent i. We define the approximate value of a path by how different it is from a path that we know agent i's true value of, i.e., a path that agent i bid on in the first stage of the mechanism. In particular we choose the path $\tilde{A}_i$ to be the 'closest' path to our solution path A_i out of all the paths that agent i bid on in the first stage of the mechanism. Our heuristic for 'closeness' is defined by the following distance

function between two bundles $A, A' \in 2^{\mathcal{Z}}$:

$$d(A, A') = \lambda||A - A'|| + (1 - \lambda)|t_A - t_A|. \tag{6.1}$$

t_A and $t_{A'}$ are the times at which paths A and A' reach the goal of agent i respectively. For paths A, A', which can each be represented by a set of points $x_1, x_2, ..., x_\ell \in \mathbb{R}^2$, we define norm $||A|| = ||x_1, x_2, \cdots, x_\ell|| = \sum_{e=0}^{f} ||x_\ell||_2$. In short, $||A - A'||$ is the sum of the Euclidean distances between the points on each path at each timestep. This distance function measures two important factors that would change an agent's value as a path changes. The first term accounts for physical locations (like rough terrain that is difficult to traverse or an area with a high number of humans that may get in the way). Paths that cross through these locations would have a higher cost. The second term accounts for a difference in flow time for the individual agent. $0 \leq \lambda \leq 1$ is thus a parameter that allows us to trade off between these two factors. With this distance function in mind, we formally define the path $\tilde{A}_i$ which is the 'closest' path to our solution path A_i (among the paths that agent i submitted as bids) as $\tilde{A}_i = \arg\min_{A' \in \text{Bids}_i} d(A_i, A')$. Finally, we define the approximate value of solution $\{A_j\}$ to agent i as $\tilde{v}_i(\{A_j\}) = b_{i,\tilde{A}_i}$.

6.3.3 Descending Auction

Once we have an approximation for the social welfare of each proposed solution in $\mathcal{AS}$, we sort $\mathcal{AS}$ in decreasing order of approximate social welfare. Then, we carry out a simultaneous descending auction to elicit the true values of each solution to each agent. Specifically, the auctioneer first offers the solution which best approximates social welfare, $\{A_j^1\}$, to each agent at $\tilde{v}_i(\{A_j^1\})$. $\tilde{v}_i(\{A_j^1\})$ is the maximum possible price agent i could be charged for being allocated $\{A_j^1\}$ at the end of the auction. If all agents accept the offer, this solution is chosen. Otherwise, the approximate value for any agent i that did not accept the offer is reset to $\tilde{v}_i(\{A_j^1\}) = \tilde{v}_i(\{A_j^1\}) - \gamma$, for some value $\gamma > 0$, and $\mathcal{AS}$ is reordered. For large γ, the auction will execute quicker, but agents are likely to be proposed offers below their value for the item. For small γ, the descending auction will occur in

shorter intervals, thus taking longer but gaining a more accurate representation of agent valuations. If at any point an agent accepted a solution, they are assumed to have accepted it in the future and are not offered a new value. If at any point an agent rejects an offer with value $\tilde{v}_i(\{A_j^1\}) < \gamma$, the offer is reset to $\tilde{v}_i(\{A_j^1\}) = 0$. If an agent rejects an offer with value $\tilde{v}_i(\{A_j^1\}) = 0$, this solution is assumed to be infeasible and removed from the set.

This process is repeated until all agents accept the same solution or all solutions have been made infeasible. If all agents accept the same solution $\{\tilde{A}_j\} \in \mathcal{AS}$, then this becomes the implemented solution and the payment for agent i is

$$p_i = \max(0, \sum_{j \in [n] \setminus \{i\}} b_{j, A_j^{-i}} - \sum_{j \in [n] \setminus \{i\}} \tilde{v}_j(\tilde{A}_j)), \tag{6.2}$$

where $\{A_j^{-i}\} \in \Gamma$ is the hypothetical solution described in Section 6.3.1 that results from a traditional VCG auction with the originally proposed bids, but without agent i. If all solutions have been deemed infeasible, the mechanism terminates with no solution.

6.3.4 Analysis

We now discuss how our modifications affect the important properties of individual rationality and incentive compatibility, and present arguments on how these are preserved by our approach.

Individual Rationality. We argue that each agent will take their assigned path. Suppose that an agent is considering deviating from their assigned path w, because there is some other path w' that has higher value for them. If w' was a path submitted to the auctioneer in the first VCG stage, they know that if it did not conflict with the other agents, they would have been allocated it. Thus, if they choose to follow w' instead of w, they are guaranteed to be in conflict with at least one other agent, which will cause w' to be infeasible. If w' was a path that was

not submitted to the auction it is *risky*, the agent has no information on whether it would cause a conflict, and thus would not choose to take it.

Incentive Compatibility. While our mechanism is not incentive compatible due to its sub-optimality [Nisan and Ronen, 2007], our design decisions prevent easy manipulation. If the algorithm does not enter the privileged knowledge sub-protocol in Section 6.3.2 then it remains in a VCG auction which is incentive compatible. If it does enter the sub-protocol, truth telling means accepting the first value that is lower than your actual value for a path. Clearly, agents are not incentivised to accept a path that is higher than their value because they may be charged for it, netting a negative utility from the process. It is also important to show they will never turn down a price that is equal to or overestimates their value. Suppose they do not, then there is a chance they will instead get offered a deal next which underestimates their value less, or a deal in which they have lower value. In the both cases, they would end up with either a worse utility solution or no solution.

6.4 Single-Agent Decision Making

Our mechanism assumes agents have the ability to submit a fixed (but small) number of bids to the auctioneer. Each bid is on a bundle of items, and it is important that an agent understands what value to assign each bundle. To better describe the MAPF problem, for the remainder of the chapter we refer to minimising the total cost, instead of maximising value, as the two problems are equivalent. If a bundle of items is not a path then it will cost an agent ∞ as it is impossible to execute. The cost of any path that eventually reaches the goal should be defined for each agent by the distance travelled and the time it takes. This will depend on the specifications of the agent, but an increase in both distance and time will be assumed to increase costs. While these costs and paths can be derived in any number of ways, and do not affect the properties of the mechanism, we outline a bidding strategy specific to the case where cost is directly linked to path length, as in a traditional MAPF setting. As we will show in Section 6.6, this method allows for a better overall result for the auction.

6.4.1 Diverse Pathfinding

We first briefly summarise some existing methods of diverse pathfinding, i.e. generating multiple, possibly suboptimal single-agent paths, as these methods are crucial to our single-agent bid generation. One way agents can find and evaluate multiple path options is through a *k-shortest path algorithm* [Yen, 1971, Eppstein, 1998]. k-shortest path algorithms can search for *simple paths*, i.e those without loops, as in Yen [1971]. Alternatively, they can search for all paths, including ones with loops and self-loops as in Eppstein [1998]. However, doing so is often very time consuming as it greatly increases the number of possible paths. There are also single-agent pathfinding algorithms that generate paths which are spatially dissimilar [Jeong and Kim, 2011, Chondrogiannis et al., 2018].

6.4.2 Single-Agent Planning

We propose a novel method for each agent to quickly generate m suboptimal paths $\{P_{rg}^1, \ldots, P_{rg}^m\}$ from a start r to a goal g, adapting the dissimilar path search algorithm from Jeong and Kim [2011] for the MAPF context. The dissimilar path search algorithm first computes the all-to-one shortest paths to g from each other vertex using Dijkstra's algorithm [Dijkstra, 1959]. It then takes the shortest path from r to g as its first selected path P_{rg}^1. We denote this path as $w = rx_1x_2 \ldots x_\ell g$. The new candidate paths are generated at each $x_d \in \{x_e\}_{e\in\{0,\ldots,\ell\}}$ (where $x_0 = r$), by branching off onto a neighbouring vertex $\tilde{x} \notin W \setminus \{x_d\}$, where W is the set of vertices in path w. Let $\tilde{x}\tilde{x_1}\tilde{x_2}...\tilde{x}_{\ell'}g$ be the shortest path between $\tilde{x}$ and g. Then the new candidate path is $\tilde{w} = rx_1x_2...x_d\tilde{x}\tilde{x_1}\tilde{x_2}...\tilde{x}_{\ell'}g$. The special case $\tilde{x} = x_d$, which was disallowed in the original algorithm but is included here, effectively incorporates wait actions into the agents' path bids, thereby providing additional flexibility to the auctioneer.

After adding the newly generated paths to the candidate set C_{rg}, a new path is selected from the set of candidates based on minimisation of a similarity metric with the currently selected paths. We introduce a similarity metric that considers both spatial and temporal similarity of two paths. To the best of our knowledge,

this is the first path generation algorithm to consider both spacial *and* temporal similarity. This metric measures the overlap of a candidate path with existing paths in both space and time:

$$P_{rg}^{b+1} = \arg\min_{P \in C_{rg}} \sum_{a=1}^{b} \frac{C\left[P_{rg}^{a} \bigcap^{t} P\right]}{C\left[P_{rg}^{a} \bigcup^{t} P\right]} \tag{6.3}$$

Here $C[\cdot]$ denotes the number of vertex-timestep pairs, and $\bigcap^t$ is a *temporal intersection* of two paths – for two paths p and q, $p \bigcap^t q$ contains all vertex-time pairs which appear in both paths, and the *temporal union* $\bigcup^t$ contains all vertex-time pairs which appear in either path. The similarity metric is designed to generate paths which do not have the agent occupying the same vertex *at the same time*, thereby providing the auctioneer with greater flexibility and improving its chances of finding a feasible allocation. The path found by Equation 6.3 is then removed from C_{rg} and used to generate new candidates, a process that is repeated until the m required paths have been selected.

6.5 Analysis

In this, section, we complete a runtime analysis of the component parts of the PKA algorithm.

1. The bid generation algorithm can be either simple or dissimilar:

   (a) The simple bid generation algorithm has a runtime of $\mathcal{O}(|E|+|V|\log(|V|)+m)$, where m is the number of paths generated [Eppstein, 1998].

   (b) The dissimilar bid generation algorithm starts with calculating the shortest path via Dijkstra for each set of start and end locations, which is $\mathcal{O}(|V|^2\log(|V|))$ [Dijkstra, 1959]. Generating each candidate path set takes time linear in the time horizon h to look up shortest paths, and m candidate sets are generated. This results in an algorithm that is $\mathcal{O}(\mathcal{O}(|V|^2\log(|V|) + h \times m)$.

Bid generation is decentralised, and thus not dependent on the number of agents.

2. The first VCG stage of the algorithm is an ILP which is exponential in the number of variables and polynomial in the number of constraints [Lenstra, 1983]. In our case, there is a variable for each bid, of which there are $m \times n$, and a constraint for each agent and each vertex-time pair. So the runtime for the first stage is $2^{\mathcal{O}((m\times n)^3)} + (h \times |V| + n)^{O(1)})$.

3. The runtime of the deconflicting stage of the auction depends on the suboptimal multi-agent pathfinding algorithm used. For an example, we consider HCA*. HCA* consists of n calls to A* over a time extended graph. Each A* instance has a worst case runtime equivalent to Dijkstra on the extended, $h \times |V|$ vertex graph, given the heuristic is admissible. The heuristic used is the Manhattan Distance over the original graph, which has a $\mathcal{O}(1)$ runtime and is admissible. Thus, the runtime of a single-agent A* run is $\mathcal{O}((h \times |V|)^2 \log(h \times |V|))$. Therefore the runtime of the multi-agent HCA* is $\mathcal{O}(n \times (h \times |V|)^2 \log(h \times |V|))$. Then, the runtime of the entire deconflicting stage, which performs HCA* f times, is $\mathcal{O}(f \times n \times (h \times |V|)^2 \log(h \times |V|))$.

4. The descending auction has a runtime of $\mathcal{O}(f)$.

We conclude the runtime for the full algorithm with simple bid generation is:

$$
\begin{aligned}
&\mathcal{O}(|E| + |V| \log(|V|) + m) \\
&\quad + \mathcal{O}(2^{\mathcal{O}((m\times n)^3)} + (h \times |V| + n)^{O(1)}) \\
&\quad + \mathcal{O}(f \times n \times (h \times |V|)^2 \log(h \times |V|)) \\
&\quad + \mathcal{O}(f) \\
&\quad = \mathcal{O}(2^{\mathcal{O}((m\times n)^3)} + f \times (h \times |V| + n)^{O(1)} + |E|)
\end{aligned}
$$

The runtime for the full algorithm with dissimilar bid generation is:

$$\begin{aligned}
&\mathcal{O}(|V|^2 \log(|V|) + h \times m) \\
&\quad + \mathcal{O}(2^{\mathcal{O}((m \times n)^3)} + (h \times |V| + n)^{O(1)}) \\
&\quad + \mathcal{O}(f \times n \times (h \times |V|)^2 \log(h \times |V|)) \\
&\quad + \mathcal{O}(f) \\
&\quad = \mathcal{O}(2^{\mathcal{O}((m \times n)^3)} + m \times f \times (h \times |V| + n)^{O(1)})
\end{aligned}$$

6.6 Evaluation

To analyse the performance of PKA, we evaluated its performance on two synthetic domains from *Dragon Age: Origins* [Stern et al., 2019] and a warehouse domain, and compared its performance to baseline methods.

6.6.1 Methods

We compare PKA to two other methods, iBundle and limited bid VCG.

1. The **iBundle** method from [Parkes, 1999] is described in Section 2.2. Agents' bidding strategies are implemented as myopic best-responses to the auction. We describe the myopic best-response in this context for clarity. At the first stage of the auction, agents bid all simple paths with length equal to their shortest path. Prices increase on current bids when agents cannot receive a valid allocation from the current bids, agents then bid the paths from the previous rounds (updated to reflect the new prices) along with new paths, which comprise of all simple paths equal to the next best path length. The agent bidding strategy is implemented using Yen's algorithm [Yen, 1971] through *NetworkX* [Hagberg et al., 2008].

2. The limited bid **VCG** auction uses the ILP combinatorial auction described in Section 2.2 with a fixed number of bids. Agents submit only 10 paths instead of bidding their entire list of possible paths, and then a traditional

VCG auction is carried out. This method is equivalent to the first stage of PKA. We consider possible types of bidding for this method:

(a) In the first method, **simple path** generation, agents submit 10 bids corresponding to their 10 shortest paths. As in iBundle bidding, these are generated with Yen's algorithm through *NetworkX*.

(b) In the second method, **dissimilar path** generation, agents submit 10 bids through the algorithm described in Section 6.4.2 using a novel similarity metric that diversifies the spread of bids from each agent over the map.

PKA, as described in Section 6.3, is implemented to request 10 bids from each agent. During the second phase of the auction, 3 possible solutions are generated with HCA* [Silver, 2005]. Because we consider MAPF scenarios where cost is equivalent to the path length, the auctioneer sets $\lambda = 0$ in Equation 6.1, i.e., it only considers time in the distance metric d. In the third stage of the auction, the price update amount γ is set to 1, as all edges in the simulated domain take time 1. PKA is also evaluated again the two types of bidding described above.

In all experiments, we directly analyse the average path costs (as opposed to social welfare) because we were evaluating on MAPF benchmarks. Note that, in this context, maximising social welfare is equivalent to minimising the sum of path costs, as shown in Amir et al. [2015] and described in Table 6.2.

All methods are implemented in Python, and ILPs (e.g., solving the winner determination problem and calculating prices for VCG and for each iBundle iteration) are solved using Gurobi. All methods generate paths as a list of vertices at a given timestep. All experiments were conducted on an AWS C5a.large EC2 instance, with 2 CPUs and 4GB of memory. We set a timeout of 300 seconds for all methods, as in Kaduri et al. [2020]. Data points were calculated over 100 trials. Full distributional data can be found in Appendix.

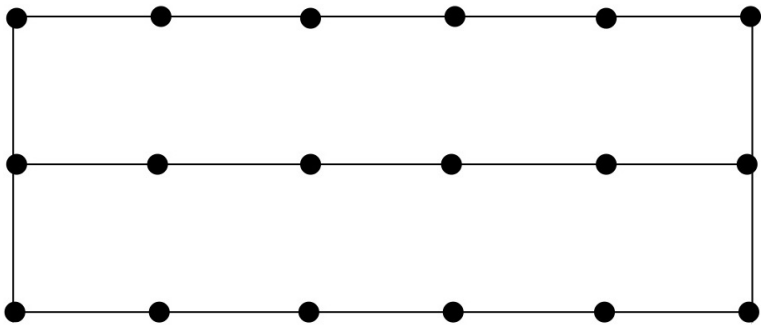


Figure 6.2: An example of a warehouse with $k = 3$ and $l = 6$.

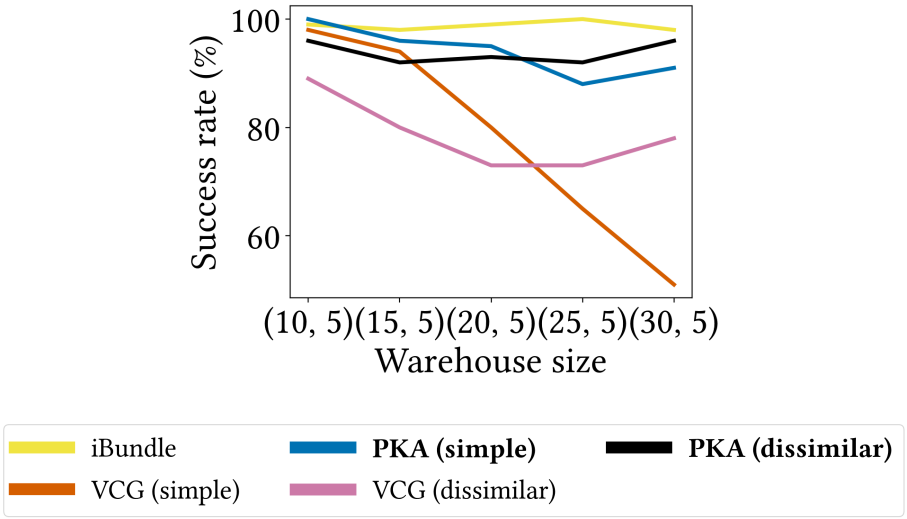


Figure 6.3: Success probabilities for a warehouse domain.

6.6.2 The Warehouse Domain

Domain Description. We first begin with a domain with few path options and high congestion, with maps designed to mirror warehouses and supermarkets. Warehouse maps consist of aisles of shelves connected to each other only at the right and left corridors. These corridors are the most likely points of conflict in the maps. All maps of size (k, l) are constructed from k aisles of length l units. An example graph $\mathcal{G}$ is shown in Figure 6.2. Start and goal vertices for each agent are generated randomly, with no two agents sharing a start or goal vertex. This domain is similar to warehouse environments used in MAPF experiments, e.g., [Ma et al., 2017, Hönig et al., 2019, Li et al., 2021a].

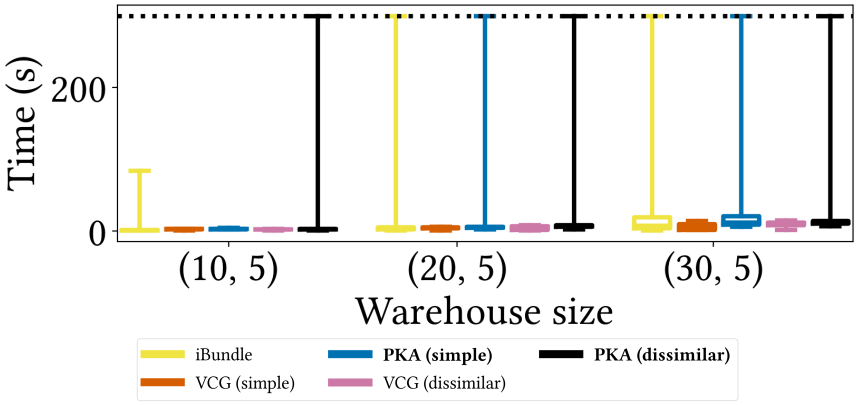


Figure 6.4: Computation time for a warehouse domain.

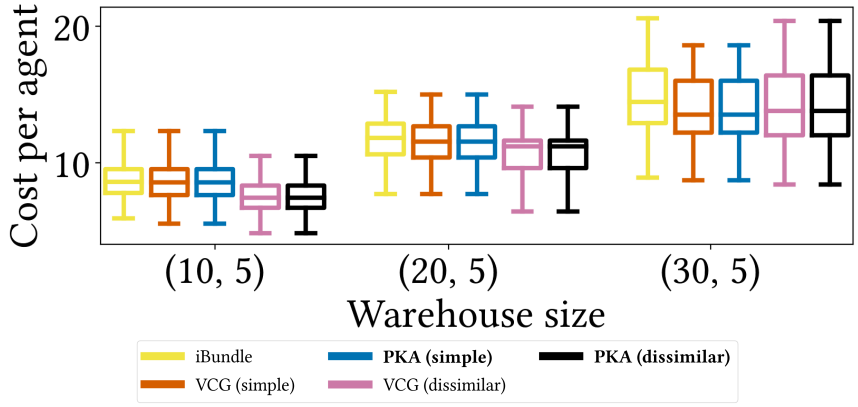


Figure 6.5: Path cost per agent for a warehouse domain.

Results. In the warehouse environment, iBundle outperforms the other methods because of the topology in the graph. With less connectivity, there are fewer valid simple paths that share the same number of edges, and thus iBundle is able to quickly achieve the optimal solution. But even in these cases, PKA, a heuristic method, is able to achieve a similar success probability as iBundle (Figure 6.3). The time of iBundle and PKA are comparable as the warehouse size increases (Figure 6.4). The cost per agent of successful trials is similar across all methods (Figure 6.5). While simple path generation is initially very successful for the same reason as iBundle, as the ratio of possible paths to submitted paths increases, the success probability of simple path generation sharply decreases. Dissimilar path generation, on the other hand, provides a good balance throughout the warehouse sizes.

6.6.3 The Dragon Age Domain

Domain Description. Next, we move to a domain with many path options to evaluate. Agents interact on two maps from *Dragon Age: Origins*, lak108d and lak110d, as shown in Figure 6.6. Both maps are drawn from a set of pathfinding baselines as described in Stern et al. [2019], and were chosen to demonstrate maps with large open areas. In these baseline domains, agents value paths exclusively based on the time it takes to reach their goal location. Only white pixels are traversable, as green represents unpassable terrain and black is out of bounds. Each pixel represents a vertex. Agents can move left, right, up, and down, which all take one timestep. Distinct start and goal vertices for each agent are generated randomly. Map lak108 has 286 vertices and map lak110 has 168 vertices.

Results. For maps lak108d and lak110d, we show the success rates of all tested methods over 100 trials in Figure 6.7. The iBundle algorithm consistently performs the worst. This is because in large, open domains like the two Dragon Age maps, agents have a large number of bids in the initial stage of the auction, as in open space there can be many possible best paths. It takes time for each agent to

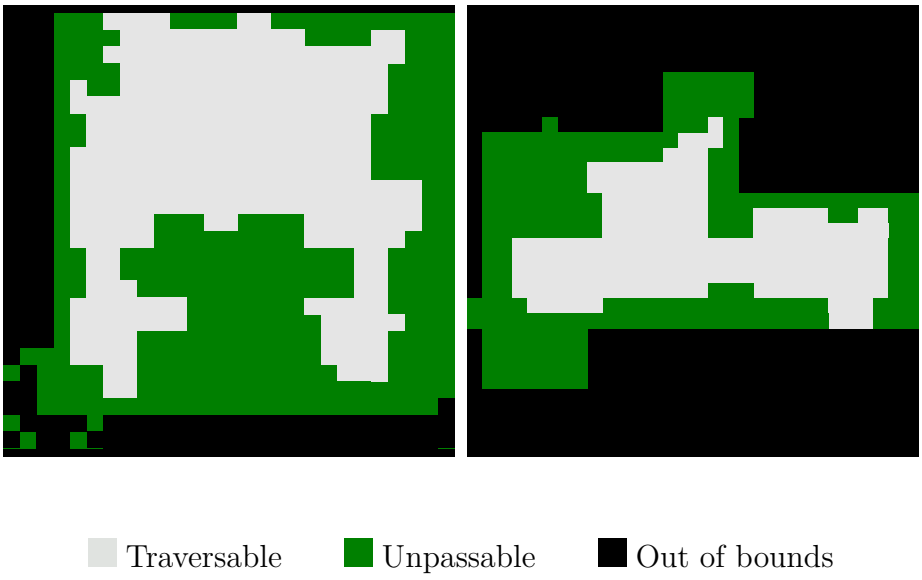


Figure 6.6: *Dragon Age: Origins* maps lak108d (left) and lak110d (right). Each pixel represents a vertex.

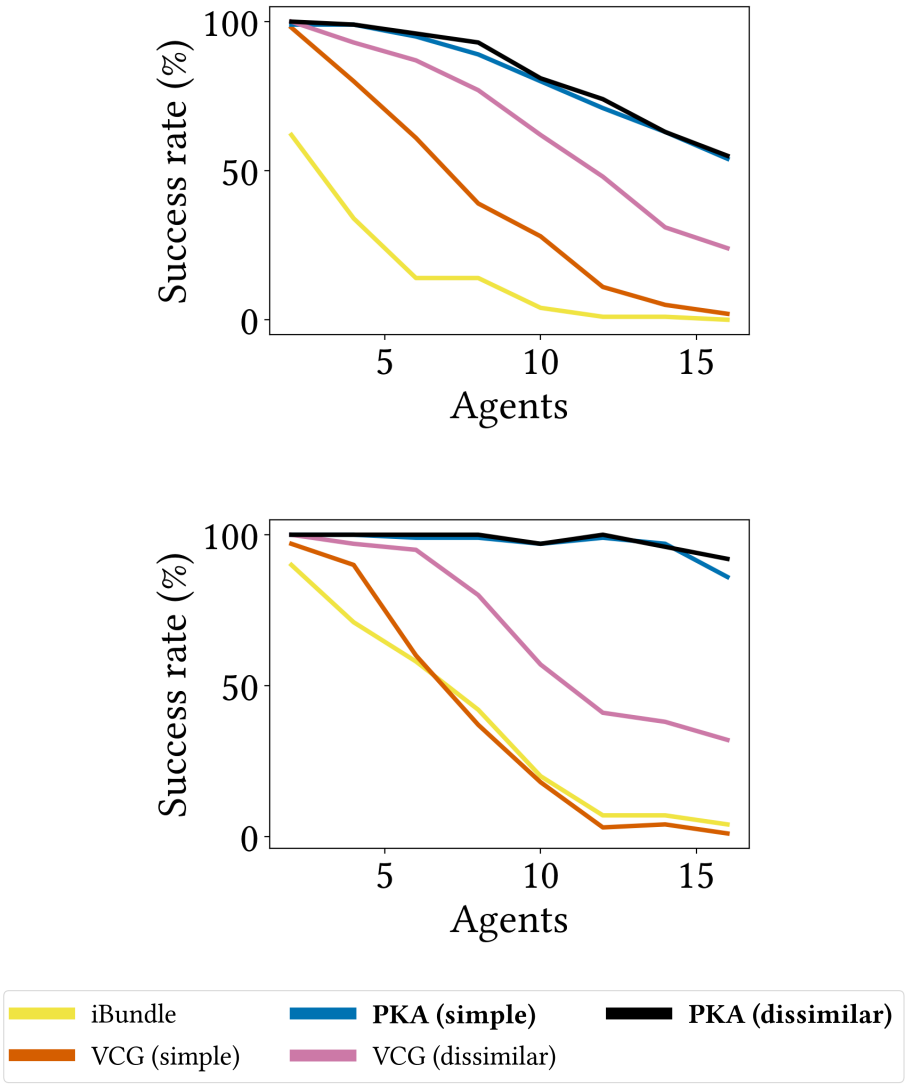


Figure 6.7: Success probabilities for *Dragon Age: Origins* maps lak108d (top) lak110d (bottom).

calculate these bids, but more importantly the large number of bids in the initial provisional allocation of iBundle can exceed the timeout of 300 seconds, as seen in Figure 6.8, and this becomes more likely as the number of agents increases. A timeout of more than 300 seconds would increase the success probability of iBundle, while a timeout of less than 300 seconds would decrease the success probability of iBundle. Figure 6.8 shows the time for each trial for every method; trials that timed out are represented in the plot as taking time equal to the timeout. By comparing VCG (simple paths) and VCG (dissimilar paths), we see the dissimilar paths are

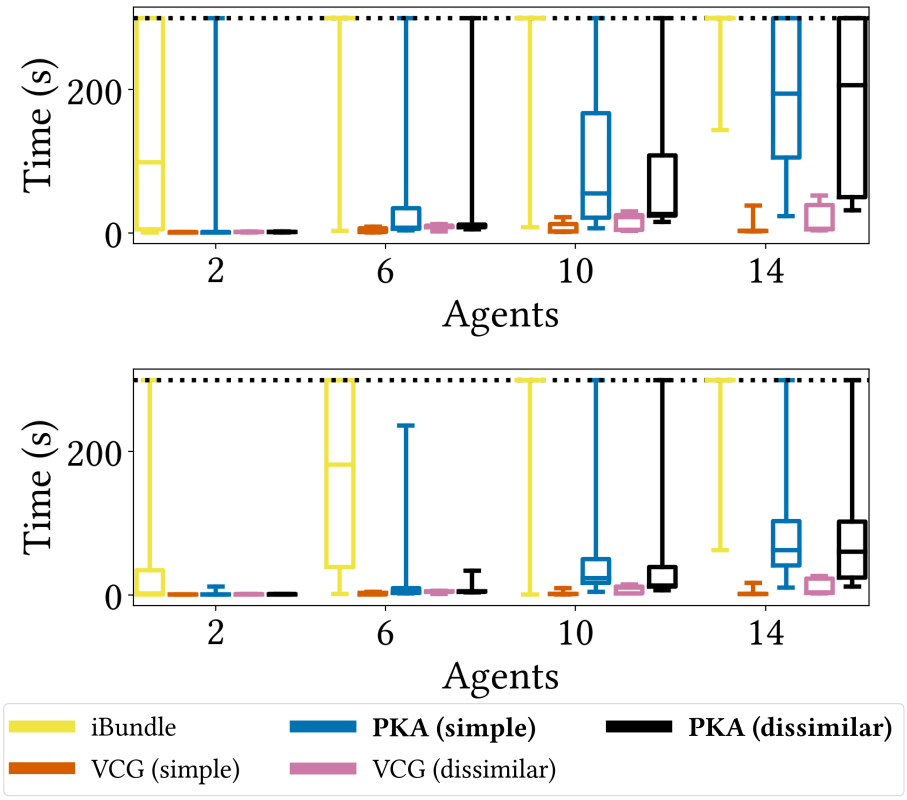


Figure 6.8: Computation time for *Dragon Age: Origins* maps lak108d (top) lak110d (bottom).

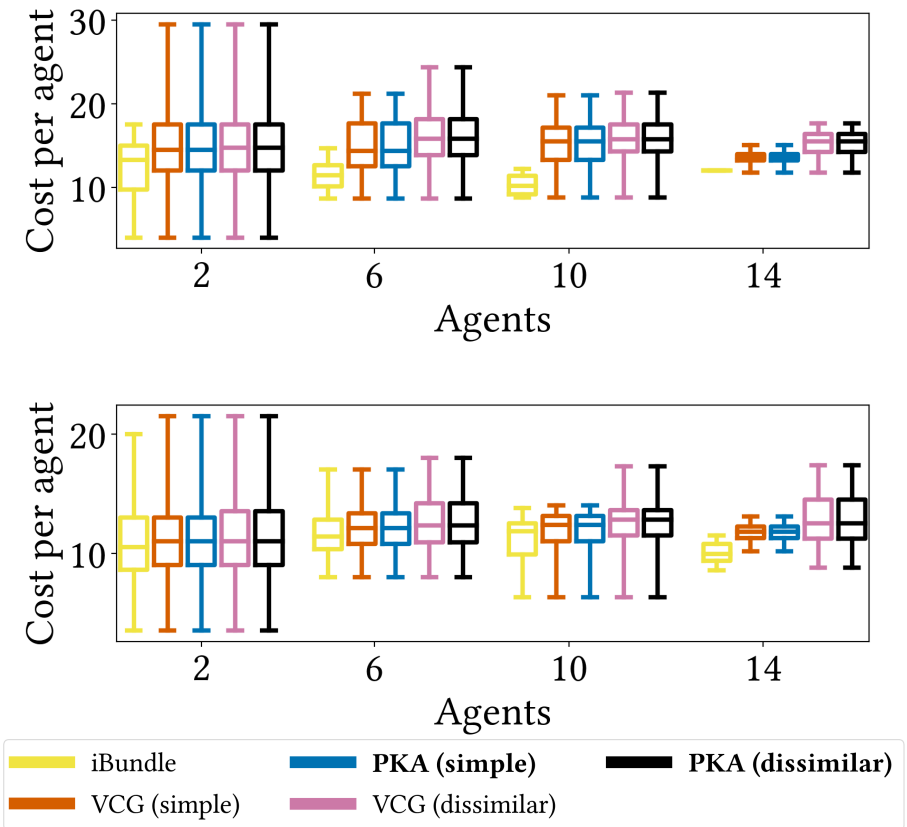


Figure 6.9: Average cost for *Dragon Age: Origins* maps lak108d (top) lak110d (bottom).

much more likely to result in a initial solution than the simple paths, and this trend persists for all numbers of agents. Because the dissimilar paths intentionally discourages space/time overlap in the submitted paths, the VCG auction has a more diverse set of options to choose from, which results in a higher chance of finding a non-conflicting solution. PKA (dissimilar paths) is more frequently successful than VCG (dissimilar paths) and PKA (simple paths) is more frequently successful than VCG (simple paths) by consistently finding acceptable solutions in the second and third stage of PKA. The cost per agent of successful trials can be seen in Figure 6.9. A successful trial is defined per method, i.e., the plot representing iBundle represents all trials where iBundle was successful. As a result, iBundle cost is lowest, as it is solving fewer instances, and the instances it is solving are those with the least amount of conflict. Similarly, across all methods the average cost path decreases as the number of agents increases, as only low conflict instances are solved.

6.7 Discussion

6.7.1 Summary

We have presented a novel method for non-cooperative multi-agent pathfinding. We extend prior work solving MAPF problems as combinatorial auctions by proposing a heuristic auction protocol that allows agents to value fewer paths by exploiting the privileged knowledge of the central planner and the physical layout of the planning space. Empirical results show that our mechanism outperforms the state-of-the-art approach for MAPF as a combinatorial auction on more general graphs.

6.7.2 Limitations and Future Work

PKA presents a good alternative to traditional VCG or iBundle in settings where there are many paths with the same length in a given graph. In such cases, PKA presents a computationally preferred alternative, but one that comes at the expense of sub optimality. One could improve the sub optimality gap of PKA by generating more possible solutions in the deconfliction stage of the mechanism, but generating too many solutions would close the computation time gap. One possible

Table 6.3: The parallel between **edge-conflict** MAPF and combinatorial auctions (CA).

MAPF	**CA**
directed edge-time pair	Item
Path (set of items)	Bundle
Path cost	Valuation function
Minimal sum of path costs	Maximal social welfare

future area of research is in generating dissimilar MAPF solutions, with similar dissimilarity metrics as we use in the single-agent dissimilar pathfinding discussed in Section 6.4.2. But generate dissimilar MAPF solutions is a difficult problem given the computational difficulty of generating just one MAPF solution. Additionally, the sub-optimality implies that the auction cannot be incentive compatible. Future work is needed to find the settings in which truthfulness is not a dominant strategy, as those settings are guaranteed to exist.

PKA is also limited by our initial problem formulation. In this work, we restricted our focus to flow time optimisation and vertex conflicts. We now discuss how our algorithm could be adapted for makespan optimisation and edge conflicts.

Makespan. To produce a MAPF solution that optimises for makespan, we would need to change the optimisation function of the combinatorial ILP (see Equation 2.10) to optimise for makespan, by instead optimising for:

$$\text{maximise } \min_{i \in n} \sum_{A \in 2^{\mathcal{Z}}} b_{i,A} x_{i,A}.$$

Note that this is phrased in terms of maximising the minimum value, which is the inverse problem to minimising the maximum cost. The suboptimal pathfinding in the deconfliction stage would not need to be changed, but in the descending auctions, solutions would be sorted by their makespan.

Edge Conflicts. PKA prevents vertex conflicts, and edge conflicts are a subset of vertex conflicts. But, because there are many vertex conflicts that are not edge conflicts, planning for vertex conflicts will likely result in a worse solution for a planner only interested in edge conflicts. To instead plan for edge conflicts, we can instead consider directed edges as a resource instead of vertices. To do this we can modify Table 6.2 and produce Table 6.3. The only other modification

required is to consider a suboptimal pathfinding algorithm in the deconfliction stage that prevents edge conflicts.

7
Conclusion

In this thesis, we have considered the problem of resource allocation in multi-agent systems. We focused on three main axes which are important for decision making surrounding resource allocation.

Allocating Resources Under Uncertainty

The first axis, allocating resources under uncertainty, was addressed with Multi-Agent Markov Decisions Processes with Constraints in Chapters 4 and 5. In Chapter 4: Allocating Chance-Constrained Resources, we considered resource allocation with a chance-constraint, which insures that, for example, that a resource limit is met by all agents 95% of the time. In Chapter 5: Allocating Risk-Constrained Resources, we considered resource allocation with a risk-constraint, which ensures that the expected value of the tail of the resource usage distribution is constrained by a resource limit.

Allocating Different Resource Types

The second axis, allocating different resource types, was studied in all chapters. But the most challenging type of allocation problem we considered, when there are multiple, time-dependent resources, was studied in Chapter 6: Allocating Space: A Non-Cooperative Approach to MAPF, where we considered allocating

space-time as a resource to find multi-agent pathfinding solutions. Space-time is a challenging category of resources to allocate because agent preferences are based on specific combinations of resources.

Allocating Resources in Non-Cooperative Settings

The third axis, non-cooperative resource allocation, was addressed in Chapters 4 and 6. In Chapter 4 we designed a price structure for ACCR combined with a lie detection mechanism to make our method applicable for non-cooperative applications. In Chapter 6 we designed a mechanism that makes use of a descending auction coupled with prices to make lying in non-cooperative auctions difficult.

We also defined a variety of questions that our thesis hoped to address. We summarise our progress below.

Q1 - Problem Definition Can we define new classes of multi-agent resource allocation problems, specifically in planning in multi-agent Markov decision processes?

In Chapter 5, we defined a novel problem formulation, the Risk-Constrained MMDP problem. The RCMMDP uses the Multi-Agent Markov Decision Process with Constraint model and considers the problem of optimising for expected reward while constraining the joint CVaR of a shared resource in a cooperative setting. While risk-constrained single-agent planning has been studied in single-agent MDPs, namely in Borkar and Jain [2014], these techniques scale poorly to multi-agent problems, as demonstrated by our experimental evaluation.

Q2 - Problem Extension Can we extend already existing multi-agent resource allocation problems to be applicable to more scenarios, in multi-agent Markov decision processes and multi-agent pathfinding?

In Chapter 4, we extend the problem of Chance-Constrained MMDPs to the non-cooperative setting, where agents' MDP information is private. We present the Auction for Chance Constrained resources, which considers how

uncertain agents are about their resource use in the bidding process with a term ϵ. We design a set of prices that are based on both traditional multi-unit VCG auctions and agent's ϵ in order to incentivise agents to truthfully report their bids. Our prices ensure that agents are not incentivised to lie in 3 out of 4 possible ways, and we provide a lie detection mechanism that can catch the 4th type of lying to ensure that agents remain truthful in expectation.

Q3 - Novel Solutions Can we find unique solution methods to the problems defined in Q1?

In Chapter 5, we defined a novel algorithm, called the Risk Contribution Approach for the Risk-Constrained MMDP problem. By leveraging the concept of risk contributions, we are able to transform the multi-agent planning problem to a series of single-agent planning problems. Our algorithm presents a signifiant computational improvement over the risk-constrained single-agent planning state-of-the-art in [Borkar and Jain, 2014], as demonstrated by our experimental evaluation.

Q4 - Algorithmic Improvement Can we find unique solution methods to the problems defined in Q3?

In Chapter 4 we demonstrate that our algorithm ACCR is not only able to solve the non-cooperative problem, but also able to outperform the cooperative baseline in many cases. The cooperative baseline from de Nijs et al. [2017] makes an approximation based on the law of large numbers, and so for up hundreds of agents ACCR is able to outperform it. This is particularly exciting because solving the non-cooperative variant of a problem is often computationally harder.

In Chapter 6 we present the Privileged Knowledge Auction, an approximation algorithm for the non-cooperative MAPF problem presented in Amir et al. [2015]. Our algorithm limits the number of bids agents need to submit to the auction to prevent the computational explosion of iBundle [Parkes, 1999], which occurs when many bundles have the same value in. Though PKA limits

the number of initial bids, it then leverages the knowledge gained in those bids combined with domain knowledge about the map to propose possible solutions which are then verified with a descending auction. In graphs with many different path options, the success rate of PKA is significantly higher than than iBundle when faced with a timeout.

Q5 - Decentralised Computation: Can we design methodology in Q3 and Q4 that decentralised the computation of multi-agent planning through novel single-agent planning methods?

In Chapter 4 we designed a novel single-agent planning technique for generating a diverse set of bids. Agents generate a Pareto front for each possible resource, with reward on one axis and ϵ on the other. Points on the Pareto front correspond to unique policies that are then submitted to the auction.

In Chapter 5 we present a single-agent algorithm from optimising for reward while bounding an agent's estimated risk contribution. Our algorithm is a modified version of Borkar and Jain [2014], which optimises for reward while bounding a risk-constraint.

In Chapter 6 we propose a new metric for dissimilarity that is tailored to the problem of allocating space-time pairs. This, in conjunction with Jeong and Kim [2011], allows for the generation of paths that are spatio-temporally dissimilar.

7.1 Limitations

While we addressed the individual limitations of our algorithms in Sections 4, 5, and 6, it is important to address the broader limitations of our framing and approach.

Weak Coupling: Throughout this thesis we consider weakly-coupled problems. The only coupling between agents is the scarce shared resource– if one agent receives a resource, that is one less resource available to the other agents. As shown in our experimental analysis in Sections 4 and 5, repeated planning over weakly-coupled single-agent models is significantly quicker than planning over the

exponentially larger joint model. But this speed up comes at the expense of expressibility– agents' actions only effect themselves, not the other agents in the system. If agents share a state space, separate from their shared resources, this assumption would not be suitable.

Pre-Allocated Policies: All methods we presented are pre-allocation methods. These allow for decentralised execution, limited communication, and individually optimal strategies for a given allocation. But the main limitation of this approach is that any failures that arise from misspecified models or out of distribution events cannot be handled online. In fully autonomous systems, this can be undesirable. Additionally, when allocating resources under uncertainty, our pre-allocation methods in Sections 4 and 5 will exceed the resource limit provided, due to the chance- or risk- constraints respectively. Online planners like [de Nijs and Stuckey, 2020], can re-plan to prevent these cases, and thus can handle more strictly constrained resources than our methods.

Finite Horizons: In all chapters, we considered finite-horizon problems. Long term goals or complex tasks may take an arbitrarily long time horizon, and more importantly that time horizon might not be known a priori. Additionally, agents may have repetitive tasks that need to be completed an infinite number of times.

Non-Cooperative Framing: Our framing of the problem as non-cooperative is relevant when agents are competing for resources, but there are many other possible relationships between agents. Adversarial relationships are often strongly coupled, and thus would require different assumptions than the ones we make in this work. Collaborative relationships can be split into two scenarios. The first, where subsets of agents are willing to either form coalitions or collude, is not covered by the non-cooperative scenarios we consider. In fact, the VCG auction we use in Chapters 4 and 6 has been shown to perform poorly in these environments [Conitzer and Sandholm, 2006]. The second scenario, cooperative relationships, is covered by the non-cooperative scenarios we consider. But, non-cooperative scenarios are more complex than cooperative scenarios, and thus often more computationally

intensive, so it is not necessarily recommend to use non-cooperative approaches in cooperative scenarios (our work in Chapter 4 is a notable exception).

7.2 Future Work

We have explored the multi-agent resource allocation problem from a variety of unique and fruitful perspectives, but much work in this area remains.

FW 1: Rich Uncertain Domains with One Resource Type in Non-Cooperative Settings. In Section 5.4, we discussed the challenges in decision making under uncertainty in non-cooperative settings with rich notions of uncertainty to even simple resources. Introducing an auction or market structure to this setting of resource allocation problems would require agents to bid with full distributional information, instead of with just a summary statistic like ϵ. This presents one interesting area of future research, and would allow us to solve for a wide variety of risk measures and notions of uncertainty. A first step toward this methodology is solving the single-agent bid generation problem of how to generate sets of policies with diverse cost distributions.

FW 2: Uncertain Domains with Many Resource Types in Non-Cooperative Settings. In Section 4.7 we discussed the challenges of expanding our methods to more different resource types. These include both instantaneous resources and multiple, combinatorial resources. As discussed in Chapter 4, our methodology could be expanded to instantaneous or multiple combinatorial resources in principle, but this would result in a combinatorial explosion in the number of Pareto fronts generated. An important step to making this tractable would be designing a search methodology that chooses which Pareto fronts to generate, instead of generating all of them. Expanding our work in Chapter 5 to multiple or instantaneous resources would require an entirely different methodology, as greedily choosing an agent whose contribution to decrease becomes more complicated when there is a distinct but coupled risk contribution for each additional timestep or resource.

FW 3: Rich Uncertain Domains with Many Resource Types in Cooperative Settings. Moving to rich uncertain domains with different resources

combines two computationally intensive tasks, planning for rich uncertainty and combinatorial preferences over resources. Work in this space will require faster single-agent methods for planning under CVaR constraints. An alternate approach would be to instead consider a risk measure like *k-of-N*, which can be efficiently approximated in MDPs [Chen and Bowling, 2012]. A first step would be to analyse the properties of *k-of-N*: Is the risk contribution associated with *k-of-N* additive like a chance constraint? Are the risk contributions approximately separable like CVaR? Additionally, because *k-of-N* considers the risk of epistemic uncertainty, an experimental analysis would require additional data sets and domains.

FW 4: Rich Uncertain Domains with Many Resource Types in Non-Cooperative Settings. The final goal of this research would be able to consider all three axes simultaneously. This requires much more research into individual intersections. The first steps would be understanding the unsolved intersections mentioned above. But additionally, the already existing approaches are computationally intensive, and the intersections will be even more so. Thus, it is important to develop better computational methods to approach the complexity of existing methods. Achieving this goal presents an important step into modelling real multi-agent applications, particularly in robotics.

FW 5: Beyond Expected Reward Throughout this thesis we focused on different constraint formulations, but in Chapters 4, 5 and 6 we consistently optimised for cumulative reward, or equivalently social welfare. An interesting area of future work would include modifying that optimisation object, for example to max-min fairness. Max-min fairness ensures that the least well-off agent is as well-off possible. In Chapters 4 and 6, our constrained optimisation methods could easily extend to max-min fairness, but the incentive structures will change under the different optimisation objective. In Chapter 5, a first step toward risk-constrained planning with multi-agent max-min fairness objectives would be new single-agent methods for the new objective.

Appendices

A Distributional Data

A.1 Distributional Data for Chapter 4

A.1.1 The Maze Domain

The following figures tables contain distributional information for the experiments on the Maze domain (Figures 4.2 and 4.3) in Section 4.6.

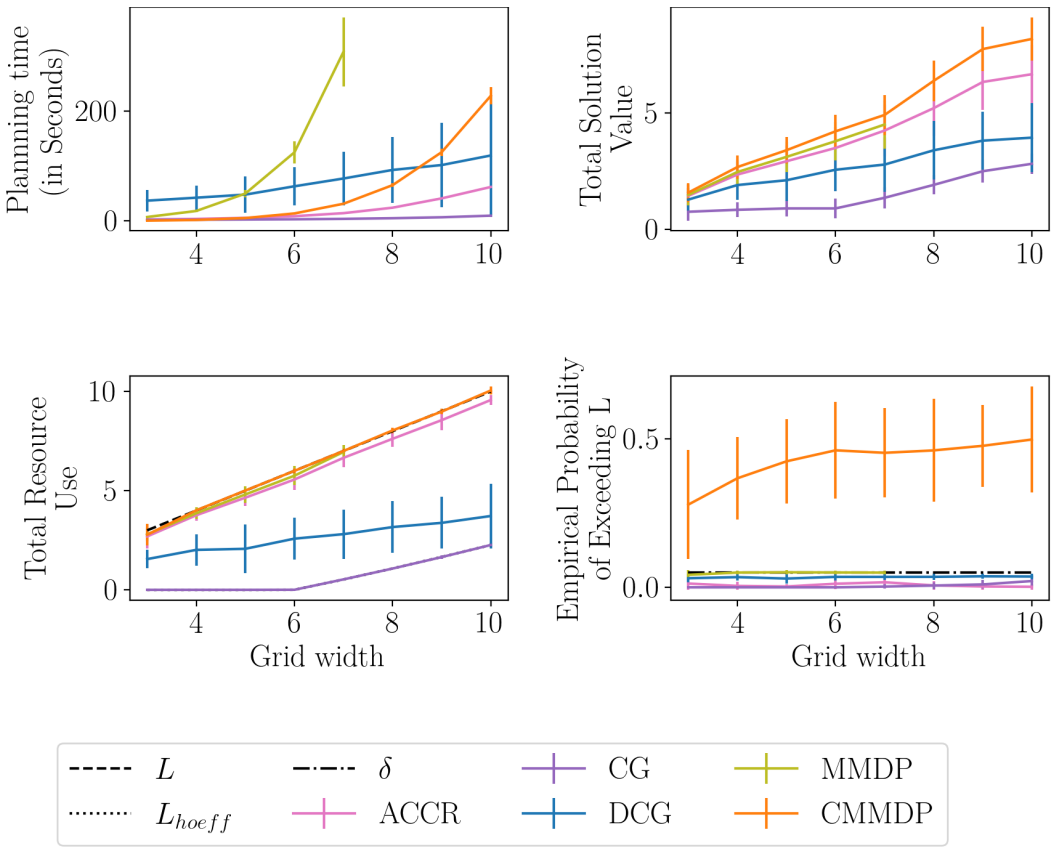


Figure A.1: Algorithm performance on Maze with increasing state space. Trials are performed with 2 agents. For an analysis of more agents, see Figure 4.3. Data points represent the average over 50 trials. This figure is identical to Figure 4.2, except has standard deviation included.

pVCG Distributional Data for Figure 4.2								
	Planning Time (in seconds)		Total Solution Value		Total Resource Use		Emp. Probability of Exceeding	
Grid Width	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.
3	2.497	0.069	1.549	0.422	2.693	0.624	0.012	0.021
4	3.278	0.239	2.314	0.594	3.762	0.292	0.004	0.012
5	5.507	1.315	2.783	0.6	4.687	0.411	0.005	0.012
6	8.513	1.273	3.596	0.803	5.634	0.436	0.011	0.016
7	13.726	1.905	3.967	0.909	6.637	0.476	0.011	0.018
8	22.705	2.27	5.114	1.08	7.627	0.365	0.005	0.014
9	40.073	4.966	6.039	1.193	8.544	0.562	0.001	0.006
10	62.122	5.579	6.923	1.102	9.466	0.491	0.004	0.011

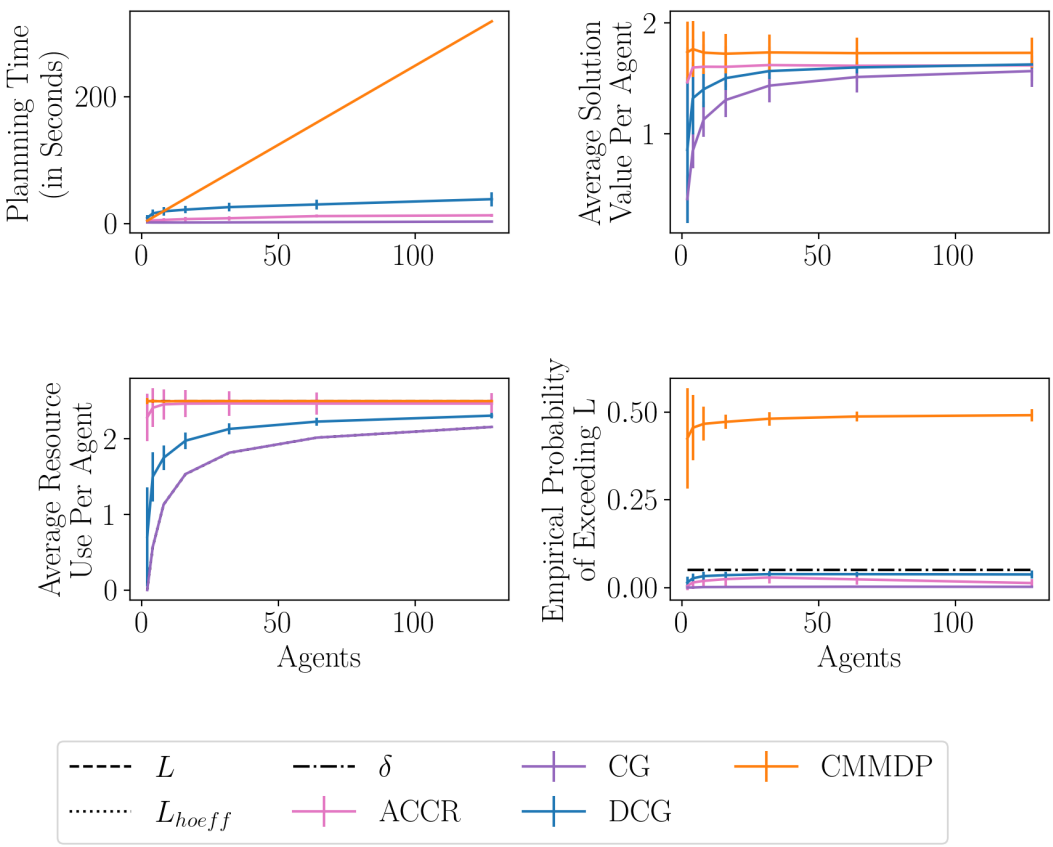


Figure A.2: Algorithm performance on Maze with increasing agents. Trials were performed with grid width 5. Data points represent the average over 50 trials. This figure is identical to Figure 4.3, except has standard deviation included.

MMDP Distributional Data for Figure 4.2								
	Planning Time (in seconds)		Total Solution Value		Total Resource Use		Emp. Probability of Exceeding	
Grid Width	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.
3	6.688	0.805	1.572	0.41	2.834	0.346	0.043	0.018
4	17.605	2.378	2.424	0.547	3.905	0.266	0.05	0.006
5	49.376	6.603	2.969	0.526	4.883	0.314	0.048	0.007
6	120.788	21.864	3.796	0.767	5.847	0.393	0.052	0.007
7	289.77	45.154	4.289	0.887	6.896	0.403	0.05	0.006

CMDP Distributional Data for Figure 4.2								
	Planning Time (in seconds)		Total Solution Value		Total Resource Use		Emp. Probability of Exceeding	
Grid Width	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.
3	0.554	0.874	1.647	0.381	2.722	0.621	0.249	0.176
4	1.84	0.148	2.581	0.535	4.005	0.062	0.318	0.119
5	5.977	0.527	3.249	0.498	4.989	0.071	0.396	0.136
6	15.996	1.517	4.185	0.695	6.022	0.118	0.437	0.152
7	36.75	3.356	4.81	0.704	7.011	0.124	0.469	0.155
8	78.049	8.183	6.304	0.691	8.028	0.172	0.503	0.154
9	149.506	16.154	7.599	0.784	8.986	0.177	0.496	0.148
10	267.393	26.711	8.291	0.858	9.975	0.19	0.447	0.129

CG Distributional Data for Figure 4.2								
	Planning Time (in seconds)		Total Solution Value		Total Resource Use		Emp. Probability of Exceeding	
Grid Width	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.
3	1.778	0.063	0.813	0.33	0.0	0.0	0.0	0.0
4	1.989	0.09	0.829	0.389	0.0	0.0	0.0	0.0
5	2.345	0.176	0.838	0.346	0.0	0.0	0.0	0.0
6	2.893	0.384	0.934	0.401	0.004	0.002	0.0	0.0
7	3.697	0.327	1.276	0.372	0.529	0.033	0.006	0.013
8	5.0	1.058	1.797	0.418	1.072	0.074	0.009	0.015
9	6.916	1.122	2.346	0.38	1.639	0.077	0.015	0.016
10	8.975	1.594	2.889	0.445	2.28	0.098	0.014	0.015

CG_d Distributional Data for Figure 4.2								
	Planning Time (in seconds)		Total Solution Value		Total Resource Use		Emp. Probability of Exceeding	
Grid Width	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.
3	32.175	17.274	1.357	0.446	1.578	0.463	0.031	0.015
4	34.112	16.989	1.88	0.663	2.112	0.801	0.034	0.011
5	31.967	12.993	1.968	0.605	1.972	0.724	0.033	0.013
6	39.251	19.327	2.385	0.913	2.324	1.146	0.034	0.012
7	54.277	39.844	2.516	1.101	2.495	1.322	0.034	0.012
8	70.904	59.531	3.063	1.261	2.864	1.548	0.037	0.011
9	81.954	59.405	3.404	1.061	2.995	1.122	0.036	0.012
10	101.292	75.768	3.99	1.405	3.764	1.548	0.035	0.013

pVCG Distributional Data for Figure 4.3								
	Planning Time (in seconds)		Average Solution Value (per agent)		Average Resource Use (per agent)		Emp. Probability of Exceeding	
Agents	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.
2	5.642	0.681	1.425	0.397	2.325	0.397	0.002	0.008
4	5.902	1.392	1.567	0.267	2.405	0.267	0.013	0.018
8	6.699	2.209	1.61	0.246	2.453	0.246	0.015	0.019
16	7.47	2.736	1.617	0.206	2.467	0.206	0.024	0.019
32	9.12	3.147	1.632	0.169	2.469	0.169	0.026	0.016
64	12.097	2.797	1.641	0.167	2.469	0.167	0.026	0.017
128	13.252	2.512	1.643	0.158	2.467	0.158	0.013	0.013

CMDP Distributional Data for Figure 4.3								
	Planning Time (in seconds)		Average Solution Value (per agent)		Average Resource Use (per agent)		Emp. Probability of Exceeding	
Agents	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.
2	5.978	0.421	1.667	0.316	2.498	0.043	0.407	0.146
4	11.809	0.809	1.719	0.258	2.499	0.023	0.425	0.097
8	23.817	2.036	1.735	0.234	2.494	0.016	0.447	0.059
16	47.29	3.353	1.734	0.198	2.501	0.011	0.47	0.021
32	94.029	6.395	1.744	0.161	2.5	0.007	0.478	0.023
64	188.181	12.341	1.751	0.159	2.501	0.006	0.485	0.014
128	377.913	26.14	1.753	0.152	2.5	0.004	0.49	0.015

CG Distributional Data for Figure 4.3								
	Planning Time (in seconds)		Average Solution Value (per agent)		Average Resource Use (per agent)		Emp. Probability of Exceeding	
Agents	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.
2	2.465	0.157	0.366	0.205	0.0	0.0	0.0	0.0
4	2.411	0.172	0.825	0.173	0.565	0.016	0.0	0.001
8	2.433	0.102	1.096	0.2	1.131	0.015	0.002	0.004
16	2.526	1.168	1.297	0.187	1.535	0.011	0.002	0.003
32	2.543	0.357	1.434	0.164	1.816	0.006	0.002	0.004
64	2.992	1.286	1.523	0.164	2.016	0.005	0.003	0.005
128	3.709	1.464	1.597	0.15	2.158	0.003	0.002	0.004

CG_d Distributional Data for Figure 4.3								
	Planning Time (in seconds)		Average Solution Value (per agent)		Average Resource Use (per agent)		Emp. Probability of Exceeding	
Agents	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.
2	10.967	6.615	0.837	0.589	0.756	0.786	0.014	0.017
4	17.884	7.304	1.265	0.326	1.493	0.349	0.025	0.016
8	20.365	5.528	1.405	0.296	1.785	0.223	0.034	0.011
16	24.697	7.098	1.492	0.248	1.98	0.151	0.039	0.007
32	27.6	6.725	1.566	0.194	2.134	0.095	0.038	0.007
64	31.133	10.143	1.615	0.184	2.23	0.069	0.039	0.007
128	39.573	11.013	1.653	0.17	2.309	0.047	0.037	0.009

A.1.2 The Autonomous Driving Domain

The following table contains distributional information for the experiments on the autonomous driving domain (Figure 4.11) in Section 4.6.

CG_d Distributional Data for Figure 4.3								
	Planning Time (in seconds)		Average Solution Value (per agent)		Average Resource Use (per agent)		Emp. Probability of Exceeding	
Agents	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.
2	1032.826	274.94	99.24	1.52	36.187	1.52	0.0	0.0
4	2029.704	352.335	97.56	4.55	39.998	4.55	0.0	0.0
6	2231.609	210.506	90.14	7.5	45.412	7.5	0.002	0.004

A.2 Distributional Data for Chapter 5

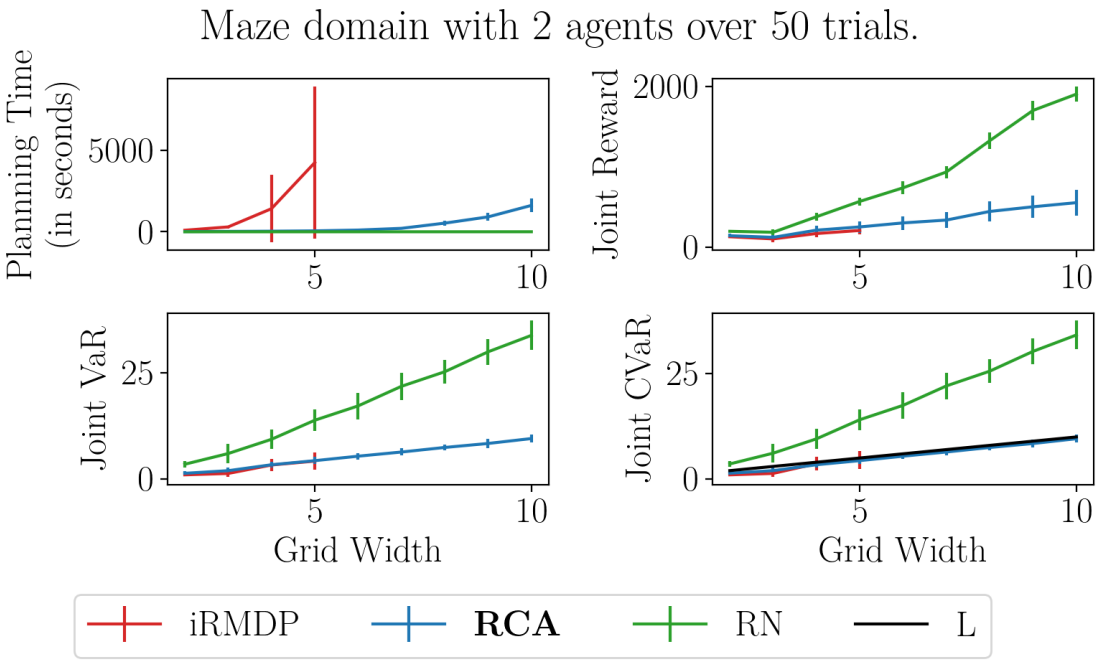


Figure A.3: An analysis of RCA and iRMDP on increasing large instances of the Maze domain for 2 agents. Each data point corresponds 50 trials. This figure is identical to Figure 5.1, except has standard deviation included.

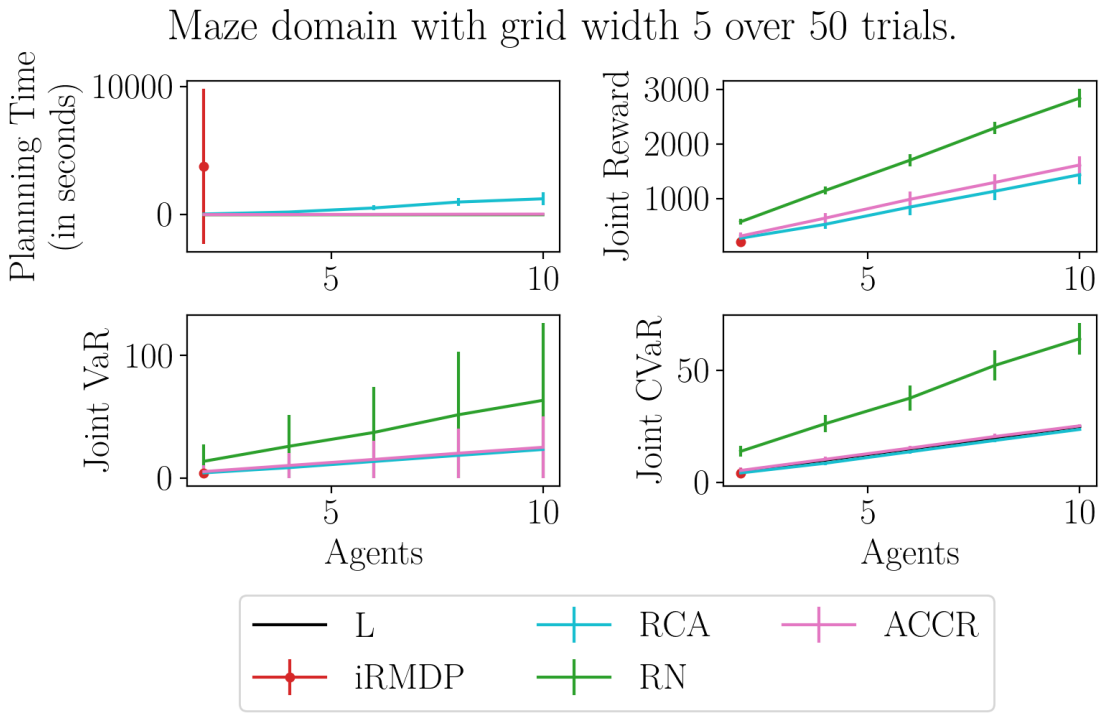


Figure A.4: An analysis of RCA and iRMDP on increasing numbers of agents in the Maze domain of gridsize 5 by 5. Each data point corresponds 50 trials. This figure is identical to Figure 5.2, except has standard deviation included.

A.2.1 The Maze Domain

The following tables contain distributional information for the experiments on the Maze domain (Figures 5.1 and 5.2) in Section 5.3.

iRMDP Distributional Data for Figure 5.1								
	Planning Time (in seconds)		Joint Reward		Joint VaR		Joint CVaR	
Grid Width	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.
2	96.108	7.557	131.804	15.226	1.0	0.0	1.0	0.0
3	299.239	92.625	105.056	42.538	1.32	0.76	1.336	0.777
4	1419.332	2066.997	171.876	45.772	3.32	1.476	3.656	1.571
5	4252.736	4684.507	209.768	49.493	4.24	1.965	4.498	2.112

RCA Distributional Data for Figure 5.1								
	Planning Time (in seconds)		Joint Reward		Joint VaR		Joint CVaR	
Grid Width	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.
2	14.742	6.12	144.806	22.003	1.36	0.48	1.387	0.482
3	25.133	11.4	124.27	37.977	1.98	0.735	2.003	0.72
4	38.457	14.468	213.664	53.159	3.38	0.66	3.4	0.641
5	58.098	19.634	253.748	71.453	4.34	0.839	4.372	0.816
6	106.612	25.073	303.596	85.223	5.4	0.748	5.431	0.718
7	212.673	60.156	338.842	95.819	6.38	0.846	6.423	0.806
8	532.119	167.531	445.752	121.833	7.46	0.754	7.498	0.709
9	916.006	252.552	503.652	136.58	8.4	1.039	8.438	1.001
10	1620.926	429.675	554.77	160.043	9.56	0.983	9.577	0.943

RN Distributional Data for Figure 5.1								
	Planning Time (in seconds)		Joint Reward		Joint VaR		Joint CVaR	
Grid Width	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.
2	0.416	0.062	199.58	0.63	3.5	0.7	3.589	0.715
3	0.147	0.023	187.458	38.009	6.02	2.258	6.144	2.266
4	0.221	0.016	379.786	47.311	9.38	2.314	9.545	2.315
5	0.354	0.057	570.308	49.636	13.88	2.535	14.087	2.529
6	0.544	0.092	739.004	80.616	17.26	3.136	17.498	3.105
7	1.055	0.185	935.676	75.119	21.88	3.173	22.117	3.129
8	1.342	0.289	1323.212	105.794	25.32	2.846	25.617	2.814
9	1.853	0.255	1699.284	119.92	30.02	3.069	30.318	3.007
10	2.686	0.34	1901.116	96.028	33.9	3.448	34.205	3.379

ACCR Distributional Data for Figure 5.1								
	Planning Time (in seconds)		Joint Reward		Joint VaR		Joint CVaR	
Grid Width	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.
2	4.683	0.153	177.848	25.927	2.3	0.458	2.332	0.473
3	4.924	0.18	149.282	45.637	3.02	0.616	3.046	0.622
4	6.484	0.306	236.704	55.522	4.32	0.989	4.461	1.053
5	10.255	0.561	293.726	79.335	5.02	0.14	5.051	0.2
6	16.736	0.948	341.94	86.832	5.96	0.28	6.07	0.408
7	27.711	1.723	390.462	106.408	7.22	0.923	7.415	1.021
8	46.425	3.593	490.994	119.028	7.98	0.14	8.117	0.514
9	75.845	8.971	583.882	131.321	9.1	1.005	9.278	1.24
10	117.703	13.158	630.428	144.016	9.98	0.14	10.083	0.654

iRMDP Distributional Data for Figure 5.2								
	Planning Time (in seconds)		Joint Reward		Joint VaR		Joint CVaR	
Grid Width	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.
2	3762.359	6048.891	208.662	47.805	3.8	1.929	4.153	2.146

RCA Distributional Data for Figure 5.2								
	Planning Time (in seconds)		Joint Reward		Joint VaR		Joint CVaR	
Grid Width	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.
2	57.751	15.506	270.408	59.646	4.26	0.82	4.291	0.795
4	213.425	92.717	525.818	90.058	8.48	0.806	8.615	0.784
6	527.34	192.882	843.448	155.838	13.52	0.755	13.722	0.757
8	990.562	302.85	1134.026	164.021	18.52	0.755	18.774	0.748
10	1244.294	495.954	1437.082	175.415	23.36	0.52	23.675	0.531

RN Distributional Data for Figure 5.2								
	Planning Time (in seconds)		Joint Reward		Joint VaR		Joint CVaR	
Grid Width	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.
2	0.964	0.667	571.93	53.259	13.7	2.394	13.912	2.385
4	0.733	0.234	1147.65	76.68	25.9	3.885	26.238	3.899
6	0.845	0.12	1705.154	114.324	37.2	5.517	37.635	5.534
8	1.266	0.296	2297.448	116.146	51.68	6.635	52.191	6.646
10	1.113	0.222	2845.35	172.204	63.48	7.148	64.066	7.172

ACCR Distributional Data for Figure 5.2								
	Planning Time (in seconds)		Joint Reward		Joint VaR		Joint CVaR	
Grid Width	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.
2	10.689	0.723	309.332	59.748	5.3	1.044	5.388	1.217
4	21.137	1.295	639.77	94.244	10.24	1.011	10.358	1.202
6	30.913	1.667	985.316	143.648	15.18	0.684	15.358	0.886
8	41.823	1.834	1295.988	154.933	20.28	0.801	20.514	1.131
10	51.992	2.582	1615.74	154.469	25.12	0.431	25.262	0.583

A.3 Distributional Data for Chapter 6

A.3.1 Warehouse Domain

The following data is from our evaluation on the *Dragon Age* domain described in Section 6.6.

Distributional Data for the Warehouse

Time Data The following three tables correspond to Figure 6.4.

Distributional Time Data for Warehouse Size (10, 5)			
	Q1	Q2	Q3
iBundle	0.453	0.747	1.215
VCG (simple)	1.984	2.381	2.676
PKA (simple)	2.017	2.443	2.718
VCG (dissimilar)	1.964	2.159	2.272
PKA (dissimilar)	2.074	2.218	2.355

Distributional Time Data for Warehouse Size (20, 5)			
	Q1	Q2	Q3
iBundle	1.644	2.53	5.226
VCG (simple)	3.725	4.494	5.158
PKA (simple)	4.345	4.985	5.64
VCG (dissimilar)	1.349	5.331	6.109
PKA (dissimilar)	5.422	6.187	7.54

Distributional Time Data for Warehouse Size (30, 5)			
	Q1	Q2	Q3
iBundle	3.573	7.099	19.256
VCG (simple)	1.176	5.943	9.185
PKA (simple)	8.902	11.493	20.092
VCG (dissimilar)	8.071	10.515	11.853
PKA (dissimilar)	10.335	11.791	13.669

Cost Data

The following three tables correspond to Figure 6.5.

Distributional Cost Data for Warehouse Size (10, 5)			
	Q1	Q2	Q3
iBundle	7.75	8.6	9.5
VCG (simple)	7.625	8.55	9.5
PKA (simple)	7.625	8.55	9.5
VCG (dissimilar)	6.7	7.4	8.3
PKA (dissimilar)	6.7	7.4	8.3

Distributional Cost Data for Warehouse Size (20, 5)			
	Q1	Q2	Q3
iBundle	10.6	11.8	12.85
VCG (simple)	10.375	11.55	12.65
PKA (simple	10.375	11.55	12.65
VCG (dissimilar)	9.6	11.2	11.6
PKA (dissimilar)	9.6	11.2	11.6

Distributional Cost Data for Warehouse Size (30, 5)			
	Q1	Q2	Q3
iBundle	12.9	14.45	16.8
VCG (simple)	12.2	13.5	16.0
PKA (simple)	12.2	13.5	16.0
VCG (dissimilar)	12.0	13.8	16.375
PKA (dissimilar)	12.0	13.8	16.375

A.3.2 *Dragon Age* Domain

The following data is from our evaluation on the *Dragon Age* domain described in Section 6.6.

Distributional Data for map lak108

Time Data The following four tables correspond to Figure 6.8 (top).

Distributional Time Data for 2 Agents			
	Q1	Q2	Q3
iBundle	4.932	98.498	300.0
VCG (simple)	0.906	1.081	1.273
PKA (simple)	0.999	1.137	1.343
VCG (dissimilar)	1.365	1.635	1.896
PKA (dissimilar)	1.46	1.685	1.945

Distributional Time Data for 6 Agents			
	Q1	Q2	Q3
iBundle	300.0	300.0	300.0
VCG (simple)	1.729	5.232	6.635
PKA (simple)	5.933	7.809	34.583
VCG (dissimilar)	7.923	9.217	10.377
PKA (dissimilar)	8.422	9.778	11.571

Distributional Time Data for 10 Agents			
	Q1	Q2	Q3
iBundle	300.0	300.0	300.0
VCG (simple)	2.176	2.385	12.949
PKA (simple)	21.123	55.214	167.368
VCG (dissimilar)	4.424	22.159	25.078
PKA (dissimilar)	24.428	26.867	108.684

Cost Data The following four tables correspond to Figure 5 (top).

Distributional Data for map lak110

Time Data The following four tables correspond to Figure 6.8 (bottom).

Cost Data The following four tables correspond to Figure 6.9 (bottom).

Distributional Time Data for 14 Agents			
	Q1	Q2	Q3
iBundle	300.0	300.0	300.0
VCG (simple)	2.867	3.124	3.397
PKA (simple)	105.508	194.273	300.0
VCG (dissimilar)	5.58	6.204	39.429
PKA (dissimilar)	49.926	205.918	300.0

Distributional Cost Data for 2 Agents			
	Q1	Q2	Q3
iBundle	9.75	13.25	15.0
VCG (simple)	12.0	14.5	17.5
PKA (simple)	12.0	14.5	17.5
VCG (dissimilar)	12.0	14.75	17.5
PKA (dissimilar)	12.0	14.75	17.5

Distributional Cost Data for 6 Agents			
	Q1	Q2	Q3
iBundle	10.083	11.417	12.667
VCG (simple)	12.5	14.333	17.667
PKA (simple)	12.5	14.333	17.667
VCG (dissimilar)	13.833	15.833	18.167
PKA (dissimilar)	13.833	15.833	18.167

Distributional Cost Data for 10 Agents			
	Q1	Q2	Q3
iBundle	9.175	10.2	11.375
VCG (simple)	13.3	15.5	17.15
PKA (simple)	13.3	15.5	17.15
VCG (dissimilar)	14.3	15.75	17.5
PKA (dissimilar)	14.3	15.75	17.5

Distributional Cost Data for 14 Agents			
	Q1	Q2	Q3
iBundle	12.0	12.0	12.0
VCG (simple)	13.214	13.643	14.0
PKA (simple)	13.214	13.643	14.0
VCG (dissimilar)	14.214	15.5	16.357
PKA (dissimilar)	14.214	15.5	16.357

Distributional Time Data for 2 Agents			
	Q1	Q2	Q3
iBundle	0.501	2.291	34.495
VCG (simple)	0.398	0.464	0.563
PKA (simple)	0.449	0.531	0.63
VCG (dissimilar)	0.635	0.765	0.926
PKA (dissimilar)	0.697	0.818	0.977

Distributional Time Data for 6 Agents			
	Q1	Q2	Q3
iBundle	39.184	181.645	300.0
VCG (simple)	0.717	2.416	3.012
PKA (simple)	2.802	3.501	9.524
VCG (dissimilar)	4.041	4.618	5.023
PKA (dissimilar)	4.26	4.717	5.133

Distributional Time Data for 10 Agents			
	Q1	Q2	Q3
iBundle	300.0	300.0	300.0
VCG (simple)	1.024	1.074	1.181
PKA (simple)	16.507	23.063	50.063
VCG (dissimilar)	2.285	9.988	11.79
PKA (dissimilar)	11.521	13.261	38.643

Distributional Time Data for 14 Agents			
	Q1	Q2	Q3
iBundle	300.0	300.0	300.0
VCG (simple)	1.349	1.429	1.513
PKA (simple)	41.27	62.632	103.148
VCG (dissimilar)	2.911	3.362	22.46
PKA (dissimilar)	23.875	60.256	101.96

Distributional Cost Data for 2 Agents			
	Q1	Q2	Q3
iBundle	8.625	10.5	13.0
VCG (simple)	9.0	11.0	13.0
PKA (simple)	9.0	11.0	13.0
VCG (dissimilar)	9.0	11.0	13.5
PKA (dissimilar)	9.0	11.0	13.5

Distributional Cost Data for 6 Agents			
	Q1	Q2	Q3
iBundle	10.333	11.417	12.792
VCG (simple)	10.792	12.083	13.333
PKA (simple)	10.792	12.083	13.333
VCG (dissimilar)	10.917	12.333	14.167
PKA (dissimilar)	10.917	12.333	14.167

Distributional Cost Data for 10 Agents			
	Q1	Q2	Q3
iBundle	9.9	11.85	12.5
VCG (simple)	10.975	12.35	13.125
PKA (simple)	10.975	12.35	13.125
VCG (dissimilar)	11.5	12.8	13.6
PKA (dissimilar)	11.5	12.8	13.6

Distributional Cost Data for 14 Agents			
	Q1	Q2	Q3
iBundle	9.357	9.929	10.786
VCG (simple)	11.268	11.786	12.214
PKA (simple)	11.268	11.786	12.214
VCG (dissimilar)	11.232	12.5	14.482
PKA (dissimilar)	11.232	12.5	14.482

Bibliography

Pritee Agrawal, Pradeep Varakantham, and William Yeoh. Scalable greedy algorithms for task/resource constrained multi-agent stochastic planning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 2016.

Mohamadreza Ahmadi, Ugo Rosolia, Michel D. Ingham, Richard M. Murray, and Aaron D. Ames. Constrained risk-averse markov decision processes. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence*, 2021.

Eitan Altman. *Constrained Markov Decision Processes: Stochastic Modeling*. Routledge, 1999.

Ofra Amir, Guni Sharon, and Roni Stern. Multi-agent pathfinding as a combinatorial auction. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

Philippe Artzner, Freddy Delbaen, Jean-Marc Eber, and David Heath. Coherent measures of risk. *Mathematical Finance*, 1999.

Lawrence M Ausubel and Peter Cramton. Auctioning securities. 1998.

Benjamin J Ayton and Brian C Williams. Vulcan: a monte carlo algorithm for large chance constrained mdps with risk bounding functions. In *Computing Research Repository*, 2018.

Steven Begley, Eric Marohn, Sabah Mikha, and Aaron Rettaliata. Digital disruption at the grocery store. *McKinsey & Company*, pages 1–8, 2020.

Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684, 1957.

Curt Bererton, Geoff Gordon, Sebastian Thrun, and Pradeep Khosla. Auction mechanism design for multi-robot coordination. In *Proceedings of the Sixteenth International Conference on Neural Information Processing Systems*, Cambridge, MA, USA, 2003. MIT Press.

Robert Bogue. Strong prospects for robots in retail. *Industrial Robot: The International Journal of Robotics Research and Application*, 46(3):326–331, 2019.

Vivek Borkar and Rahul Jain. Risk-constrained markov decision processes. *Transactions on Automatic Control*, 59(9):2574–2579, 2014.

Craig Boutilier. Planning, learning and coordination in multiagent decision processes. In *Proceedings of the Sixth Conference on Theoretical Aspects of Rationality and Knowledge*, 1996.

Craig Boutilier and Tyler Lu. Budget allocation using weakly coupled, constrained markov decision processes. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, 2016.

Ronen I. Brafman, Carmel Domshlak, Yagil Engel, and Moshe Tennenholtz. Planning games. In *Proceedings of the Twenty First International Joint Conference on Artificial Intelligence*, 2009.

Jesus Capitan, Matthijs T.J. Spaan, Luis Merino, and Anibal Ollero. Decentralized multi-robot cooperation with auctioned pomdps. *Journal of Artificial Intelligence Research*, 32(6):650–671, 2013.

Rohan Chandra, Rahul Maligi, Arya Anantula, and Joydeep Biswas. Socialmapf: Optimal and efficient multi-agent path finding with strategic agents for social navigation. In *Proceedings from the AAAI-23 Workshop on Multi-Agent Path Finding*, 2022.

Katherine Chen and Michael Bowling. Tractable objectives for robust policy optimization. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Proceedings of the Twenty-Fifth Advances in Neural Information Processing Systems*, 2012.

Theodoros Chondrogiannis, Panagiotis Bouros, Johann Gamper, Ulf Leser, and David B. Blumenthal. Finding k-dissimilar paths with minimum collective length. In *Proceedings of the Twenty-Sixth ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2018.

Yinlam Chow and Mohammad Ghavamzadeh. Algorithms for cvar optimization in mdps. In *Proceedings of the Twenty-Seventh International Conference on Neural Information Processing Systems*, 2014.

Yinlam Chow, Aviv Tamar, Shie Mannor, and Marco Pavone. Risk-sensitive and robust decision-making: a cvar optimization approach. In *Proceedings of the Twenty-Eighth International Conference on Neural Information Processing Systems*, 2015.

Yinlam Chow, Mohammad Ghavamzadeh, Lucas Janson, and Marco Pavone. Risk-constrained reinforcement learning with percentile risk criteria. *The Journal of Machine Learning Research*, 18(1):6070–6120, 2017.

Edward H. Clarke. Multipart pricing of public goods. *Public Choice*, 11:17–33, 1971.

Vincent Conitzer and Tuomas Sandholm. Failures of the vcg mechanism in combinatorial auctions and exchanges. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 521–528, 2006.

Peter Cramton, Yoav Shoham, and Richard Steinberg. *Combinatorial Auctions*. The MIT Press, Cambridge, MA, USA, 2006.

Frits de Nijs and Peter J Stuckey. Risk-aware conditional replanning for globally constrained multi-agent sequential decision making. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pages 303–311, 2020.

Frits De Nijs, Matthijs Spaan, and Mathijs de Weerdt. Best-response planning of thermostatically controlled loads under power constraints. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

Frits de Nijs, Erwin Walraven, Mathijs de Weerdt, and Matthijs Spaan. Bounding the probability of resource constraint violations in multi-agent mdps. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Frits de Nijs, Mathijs M De Weerdt, and Matthijs TJ Spaan. Multi-agent planning under uncertainty for capacity management. *Intelligent Integrated Energy Systems: The PowerWeb Program at TU Delft*, pages 197–213, 2019.

Frits de Nijs, Erwin Walraven, Mathijs De Weerdt, and Matthijs Spaan. Constrained multiagent markov decision processes: A taxonomy of problems and algorithms. *Journal of Artificial Intelligence Research*, 70:955–1001, 2021.

Sven De Vries and Rakesh V Vohra. Combinatorial auctions: A survey. *INFORMS Journal on computing*, 15(3):284–309, 2003.

Christian Dehnert, Sebastian Junges, Joost-Pieter Katoen, and Matthias Volk. A storm is coming: A modern probabilistic model checker. In *Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part II 30*, pages 592–600. Springer, 2017.

Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

Shahar Dobzinski and Noam Nisan. Limitations of vcg-based mechanisms. In *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*, 2007.

Shahar Dobzinski and Noam Nisan. Mechanisms for multi-unit auctions. *Journal of Artificial Intelligence Research*, 37:85–98, 2010.

Dmitri A. Dolgov and Edmund H. Durfee. Resource allocation among agents with mdp-induced preferences. *Journal of Artificial Intelligence Research*, 27:505–549, 2006.

David Eppstein. Finding the k shortest paths. *SIAM Journal on Computing*, 28(2): 652–673, 1998.

Kousha Etessami, Marta Kwiatkowska, Moshe Y Vardi, and Mihalis Yannakakis. Multi-objective model checking of markov decision processes. In *Proceedings of the Thirteenth International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 2007.

Amal Feriani and Ekram Hossain. Single and multi-agent deep reinforcement learning for ai-enabled wireless networks: A tutorial. *IEEE Communications Surveys & Tutorials*, 23(2):1226–1252, 2021.

Vojtěch Forejt, Marta Kwiatkowska, and David Parker. Pareto curves for probabilistic model checking. In *Proceedings of the Tenth International Conference on Automated Technology for Verification and Analysis*, 2012.

Anna Gautier, Alex Stephens, Bruno Lacerda, Nick Hawes, and Michael Wooldridge. Negotiated path planning for non-cooperative multi-robot systems. In *Proceedings of the Twenty-First International Conference on Autonomous Agents and Multiagent Systems*, 2022.

Anna Gautier, Bruno Lacerda, Nick Hawes, and Michael Wooldridge. Multi-unit auctions for allocating chance-constrained resources. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence*, 2023a.

Anna Gautier, Marc Rigter, Bruno Lacerda, Nick Hawes, and Michael Wooldridge. Risk-constrained planning for multi-agent systems with shared resources. In

Proceedings of the Twenty-Second International Conference on Autonomous Agents and Multiagent Systems, 2023b.

Brian P. Gerkey and Maja J. Mataric. Sold!: Auction methods for multirobot coordination. *IEEE Transactions on Robotics and Automation*, 18(5):758–768, 2002.

Jennifer Gielis, Ajay Shankar, and Amanda Prorok. A critical review of communications in multi-robot systems. *Current Robotics Reports*, 3(4):213–225, 2022.

Alessandro Giuseppi and Antonio Pietrabissa. Chance-constrained control with lexicographic deep reinforcement learning. *IEEE Control Systems Letters*, 4(3): 755–760, 2020.

Theodore Groves. Incentives in teams. *Econometrica*, 41(4):617–631, 1973.

Geordan Gutow and Jonathan D. Rogers. And/or search techniques for chance constrained motion primitive path planning. *Robotics and Autonomous Systems*, 2021.

Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using networkx. In *Proceedings of the Seventh Python in Science Conference*, 2008.

William B. Haskell and Rahul Jain. A convex analytic approach to risk-aware markov decision processes. *SIAM Journal on Control and Optimization*, 53(3): 1569–1598, 2015.

Takuya Hiraoka, Takahisa Imagawa, Tatsuya Mori, Takashi Onishi, and Yoshimasa Tsuruoka. Learning robust options by conditional value at risk optimization. In *Proceedings of the Thirty-Third International Conference on Neural Information Processing Systems*, 2019.

Xin Huang, Ashkan Jasour, Matthew Deyo, Andreas Hofmann, and Brian C. Williams. Hybrid risk-aware conditional planning with applications in autonomous vehicles. In *Proceedings of the 2018 Conference on Decision and Control*, 2018.

Luke Hunsberger and Barbara J. Grosz. A combinatorial auction for collaborative planning. In *Proceedings Fourth International Conference on MultiAgent Systems*, 2000.

Leonid Hurwicz and Stanley Reiter. *Designing economic mechanisms*. Cambridge University Press, 2006.

Wolfgang Hönig, T.K. Satish Kumar, Liron Cohen, Hang Ma, Hong Xu, Nora Ayanian, and Sven Koenig. Multi-agent path finding with kinematic constraints. In *Proceedings of the Twenty-Sixth International Conference on Automated Planning and Scheduling*, 2016.

Wolfgang Hönig, Scott Kiesel, Andrew Tinka, Joseph W. Durham, and Nora Ayanian. Persistent and robust execution of mapf schedules in warehouses. *IEEE Robotics and Automation Letters*, 4(2):1125–1131, 2019.

Samuel Ieong, Mukund Sundararajan, and Anthony Man-Cho So. Mechanism design for stochastic optimization problems. *SIGecom Exchanges*, page 52–54, 2007.

Dylan Jennings and Miguel Figliozzi. Study of sidewalk autonomous delivery robots and their potential impacts on freight efficiency and travel. *Transportation Research Record*, 2673(6):317–326, 2019.

Yeonjeong Jeong and Dong-Kyu Kim. Dissimilar alternative path search algorithm using a candidate path set. In *Search Algorithms and Applications*, page 409. IntechOpen, Rijeka, Croatia, 2011.

Omri Kaduri, Eli Boyarski, and Roni Stern. Algorithm selection for optimal multi-agent pathfinding. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pages 161–165, 2020.

Michael Kearns, Yishay Mansour, and Satinder Singh. Fast planning in stochastic games. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, 2000.

Ramtin Keramati, Christoph Dann, Alex Tamkin, and Emma Brunskill. Being optimistic to be conservative: Quickly learning a cvar policy. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

Zeashan Hameed Khan, Afifa Siddique, and Chang Won Lee. Robotics utilization for healthcare digitization in global covid-19 management. *International Journal of Environmental Research and Public Health*, 17(11):3819, 2020.

Sven Koenig, C Tovey, M Lagoudakis, V Markakis, David Kempe, Pinar Keskinocak, A Kleywegt, Adam Meyerson, and Sonal Jain. The power of sequential single-item auctions for agent coordination. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*, 2006.

Vijay Krishna. *Auction Theory*. Elsevier Academic Press, Boston, 2nd edition, 2009.

Anthony M Kwasnica and Katerina Sherstyuk. Multiunit auctions. *A Collection of Surveys on Market Experiments*, pages 75–108, 2013.

Marta Kwiatkowska, Gethin Norman, and David Parker. Prism: Probabilistic symbolic model checker. In *Proceedings of the Twelfth International Conference on Computer Performance Evaluation, Modelling Techniques and Tools*, 2002.

Maria Kyrarini, Fotios Lygerakis, Akilesh Rajavenkatanarayanan, Christos Sevastopoulos, Harish Ram Nambiappan, Kodur Krishna Chaitanya, Ashwin Ramesh Babu, Joanne Mathew, and Fillia Makedon. A survey of robots in healthcare. *Technologies*, 9(1):8, 2021.

Prashanth L. A. and Michael C. Fu. Risk-sensitive reinforcement learning via policy gradient search. *Foundations and Trends Machine Learning*, 15(5):537–693, 2022.

Michail G. Lagoudakis, Evangelos Markakis, David Kempe, Pinar Keskinocak, Anton J. Kleywegt, Sven Koenig, Craig A. Tovey, Adam Meyerson, and Sonal Jain. Auction-based multi-robot routing. In *Proceeeds of the 2005 Robotics: Science and Systems*, 2005.

Daniel Lehmann, Rudolf Müller, and Tuomas Sandholm. The winner determination problem. *Combinatorial Auctions*, pages 297–318, 2006.

Hendrik W Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, 1983.

Chaojie Li, Yan Xu, Xinghuo Yu, Caspar Ryan, and Tingwen Huang. Risk-averse energy trading in multienergy microgrids: A two-stage stochastic game approach. *IEEE Transactions on Industrial Informatics*, 13(5):2620–2630, 2017.

Jiaoyang Li, Andrew Tinka, Scott Kiesel, Joseph W. Durham, T. K. Satish Kumar, and Sven Koenig. Lifelong multi-agent path finding in large-scale warehouses. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(13):11272–11281, May 2021a.

Xiaocheng Li, Huaiyang Zhong, and Margaret L Brandeau. Quantile markov decision processes. *Operations Research*, 70(3):1428–1447, 2021b.

Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the Eleventh International Conference on International Conference on Machine Learning*, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

Johan Los, Frederik Schulte, Margaretha Gansterer, Richard F Hartl, Matthijs TJ Spaan, and Rudy R Negenborn. Decentralized combinatorial auctions for dynamic and large-scale collaborative vehicle routing. In *Proceedings of the Eleventh International Conference on Computational Logistics*, pages 215–230, 2020.

Hang Ma, T. K. Satish Kumar, and Sven Koenig. Multi-agent path finding with delay probabilities. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Rajiv T Maheswaran and Tamer Başar. Nash equilibrium and decentralized negotiation in auctioning divisible resources. *Group Decision and Negotiation*, 12 (5):361–395, 2003.

Mitchell McIntire, Ernesto Nunes, and Maria Gini. Iterated multi-robot auctions for precedence-constrained task scheduling. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, 2016.

Nicolas Meuleau, Milos Hauskrecht, Kee-Eung Kim, Leonid Peshkin, Leslie Pack Kaelbling, Thomas L Dean, and Craig Boutilier. Solving very large weakly coupled markov decision processes. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, 1998.

Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Cambridge University Press, 1995.

Roger B Myerson. *Mechanism design*. Springer, 1989.

Noam Nisan and Amir Ronen. Computationally feasible vcg mechanisms. *Journal of Artificial Intelligence Research*, 29:19–47, 2007.

Luyao Niu and Andrew Clark. Optimal secure control with linear temporal logic constraints. *IEEE Transactions on Automatic Control*, 65(6):2434–2449, 2019.

Ernesto Nunes and Maria Gini. Multi-robot auctions for allocation of tasks with temporal constraints. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

Christos H Papadimitriou and Mihalis Yannakakis. On the approximability of trade-offs and optimal access of web sources. In *Proceedings Fourty-First Annual Symposium on Foundations of Computer Science*, pages 86–92, 2000.

David C. Parkes. ibundle: An efficient ascending price bundle auction. In *Proceedings of the First ACM Conference on Electronic Commerce*, page 148–157, 1999.

Martin L Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2014.

Wei Qiu, Xinrun Wang, Runsheng Yu, Rundong Wang, Xu He, Bo An, Svetlana Obraztsova, and Zinovi Rabinovich. Rmix: Learning risk-sensitive policies for cooperative reinforcement learning agents. In *Proceedings of the Thirty-Fourth International Conference on Neural Information Processing Systems*, 2021.

Sarvapali D Ramchurn, Feng Wu, Wenchao Jiang, Joel E Fischer, Steve Reece, Stephen Roberts, Tom Rodden, Chris Greenhalgh, and Nicholas R Jennings. Human–agent collaboration for disaster response. *Autonomous Agents and Multi-Agent Systems*, 30:82–111, 2016.

Marc Rigter, Bruno Lacerda, and Nick Hawes. Risk-averse bayes-adaptive reinforcement learning. In *Proceedings of the Thirty-Fourth International Conference on Neural Information Processing Systems*, 2021.

Marc Rigter, Paul Duckworth, Bruno Lacerda, and Nick Hawes. Planning for risk-aversion and expected value in mdps. In *Proceedings of the Thirty-Second International Conference on Automated Planning and Scheduling*, 2022.

R Tyrrell Rockafellar and Stanislav Uryasev. Optimization of conditional value-at-risk. *Journal of Risk*, 2:21–42, 2000.

Diederik M Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, 48:67–113, 2013.

Pedro Santana, Sylvie Thiébaux, and Brian Williams. Rao*: An algorithm for chance-constrained pomdp's. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

Sergey Sarykalin, Gaia Serraino, and Stan Uryasev. Value-at-risk vs. conditional value-at-risk in risk management and optimization. In *State-of-the-Art Decision-Making Tools in the Information-Intensive Age*, pages 270–294. Informs, 2008.

Bharadwaj Satchidanandan and Munther A. Dahleh. Incentive compatibility in two-stage repeated stochastic games. Preprint, 2022.

Philipp Schillinger, Mathias Bürger, and Dimos V. Dimarogonas. Auctioning over probabilistic options for temporal logic-based multi-robot cooperation under uncertainty. In *Proceedings of the 2018 International Conference on Robotics and Automation*, 2018.

Tomer Shahar, Shashank Shekhar, Dor Atzmon, Abdallah Saffidine, Brendan Juba, and Roni Stern. Safe multi-agent pathfinding with time uncertainty. *Journal of Artificial Intelligence Research*, 70:923–954, 2021.

Guni Sharon, Roni Stern, Ariel Felner, and Nathan R Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219:40–66, 2015.

David Silver. Cooperative pathfinding. In *Proceedings of the First AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2005.

Eilon Solan and Nicolas Vieille. Stochastic games. *Proceedings of the National Academy of Sciences*, 112(45):13743–13746, 2015.

Trevor Standley. Finding optimal solutions to cooperative pathfinding ppoblems. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

Roni Stern, Nathan R. Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne T. Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, T. K. Satish Kumar, Eli Boyarski, and Roman Bartak. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *Proceedings of the 2019 Symposium on Combinatorial Search*, pages 151–158, 2019.

Charlie Street. *Multi-Robot Coordination Under Temporal Uncertainty.* PhD thesis, University of Oxford, 2022. Chapter 7.

Charlie Street, Sebastian Pütz, Manuel Mühlig, Nick Hawes, and Bruno Lacerda. Congestion-aware policy synthesis for multirobot systems. *IEEE Transactions on Robotics*, pages 1–19, 2021.

Aviv Tamar, Yinlam Chow, Mohammad Ghavamzadeh, and Shie Mannor. Policy gradient for coherent risk measures. In *Proceedings of the Twenty-Eighth International Conference on Neural Information Processing Systems*, 2015a.

Aviv Tamar, Yonatan Glassner, and Shie Mannor. Optimizing the cvar via sampling. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015b.

Dirk Tasche. Risk contributions and performance measurement. *Report of the Lehrstuhl für mathematische Statistik, TU München*, 1999.

Eric M. Timmons, Marlyse Reeves, Benjamin J. Ayton, Brian Williams, Michel D. Ingham, Julie Castillo-Rogez, William Seto, Klaus Havelund, Ashkan Jasour, Benjamin Donitz, Declan Mages, Amir Rahmani, Peyman Tavallali, and Seung Chung. Information-driven and risk-bounded autonomy for scientist avatars. In *Proceedings of ASCEND*, 2021.

Craig Tovey, Michail G Lagoudakis, Sonal Jain, and Sven Koenig. The generation of bidding rules for auction-based robot coordination. In *Proceedings from the 2005 International Workshop on Multi-Robot Systems*, pages 3–14, 2005.

William Vickrey. Counterspeculation, Auctions, and Competitive Sealed Tenders. *The Journal of Finance*, 16(1):8–37, 1961.

Glenn Wagner and Howie Choset. Path planning for multiple agents under uncertainty. In *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling*, 2017.

Thayne T. Walker, David M. Chan, and Nathan R. Sturtevant. Using hierarchical constraints to avoid conflicts in multi-agent pathfinding. In *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling*, 2017.

Erwin Walraven and Matthijs TJ Spaan. Column generation algorithms for constrained pomdps. *Journal of Artificial Intelligence Research*, 62:489–533, 2018.

Catherine Wolfram. Strategic bidding in a multi-unit auction: An empirical analysis of bids to supply electricity, 1997.

Laurence A Wolsey. *Integer Programming*. John Wiley & Sons, 2020.

Jianhui Wu and Edmund H Durfee. Resource-driven mission-phasing techniques for constrained agents in stochastic environments. *Journal of Artificial Intelligence Research*, 38:415–473, 2010.

Yasuhiro Yamai and Toshinao Yoshiba. Comparative analyses of expected shortfall and value-at-risk: Their estimation error, decomposition, and optimization. *Monetary and Economic Studies*, 20(1):87–121, 2002.

Jin Y Yen. Finding the k shortest loopless paths in a network. *Management Science*, 17(11):712–716, 1971.

Chengyang Ying, Xinning Zhou, Hang Su, Dong Yan, Ning Chen, and Jun Zhu. Towards safe reinforcement learning via constraining conditional value-at-risk. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, 2022.

Kirk A Yost and Alan R Washburn. The lp/pomdp marriage: Optimization with imperfect information. *Naval Research Logistics*, 47(8):607–619, 2000.

Chao Yu, Xin Wang, Xin Xu, Minjie Zhang, Hongwei Ge, Jiankang Ren, Liang Sun, Bingcai Chen, and Guozhen Tan. Distributed multiagent coordinated learning for autonomous driving in highways based on dynamic coordination graphs. *Ieee transactions on intelligent transportation systems*, 21(2):735–748, 2019.

Pengqian Yu, William B. Haskell, and Huan Xu. Approximate value iteration for risk-aware markov decision processes. *Transactions on Automatic Control*, 63(9): 3135–3142, 2018.

Won Joon Yun, Soohyun Park, Joongheon Kim, MyungJae Shin, Soyi Jung, David A. Mohaisen, and Jae-Hyun Kim. Cooperative multiagent deep reinforcement learning for reliable surveillance via autonomous multi-uav control. *IEEE Transactions on Industrial Informatics*, 18(10):7086–7096, 2022.