Risk-Constrained Planning for Multi-Agent Systems with Shared Resources

Anna Gautier
University of Oxford
Oxford, United Kingdom
anna.gautier@eng.ox.ac.uk

Marc Rigter
University of Oxford
Oxford, United Kingdom
mrigter@robots.ox.ac.uk

Bruno Lacerda
University of Oxford
Oxford, United Kingdom
bruno@robots.ox.ac.uk

Nick Hawes
University of Oxford
Oxford, United Kingdom
nickh@robots.ox.ac.uk

Michael Wooldridge
University of Oxford
Oxford, United Kingdom
mjw@cs.ox.ac.uk

ABSTRACT

Planning under uncertainty requires complex reasoning about future events, and this complexity increases with the addition of multiple agents. One problem faced when considering multi-agent systems under uncertainty is the handling of shared resources. Adding a resource constraint limits the actions that agents can take, forcing collaborative decision making on who gets to use what resources. Prior work has considered different formulations, such as satisfying a resource constraint in expectation or ensuring that a resource constraint is met some percent of the time. However, these formulations of constrained planning ignore important distributional information about resource usage. Namely, they do not consider how bad the worst cases can get. In this paper, we formulate a *risk-constrained* shared resource problem and aim to limit the risk of excessive use of such resources. We focus on optimising for reward while constraining the Conditional Value-at-Risk (CVaR) of the shared resource. 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. We present an algorithm that uses risk contributions to iteratively update single-agent policies until the joint risk constraint is satisfied. We evaluate our algorithm on two synthetic domains.

CCS CONCEPTS

• **Computing methodologies** → **Planning under uncertainty**; **Multi-agent planning**.

KEYWORDS

multi-agent systems; planning under uncertainty; risk-aware planning; conditional value-at-risk; risk contribution

ACM Reference Format:
Anna Gautier, Marc Rigter, Bruno Lacerda, Nick Hawes, and Michael Wooldridge. 2023. Risk-Constrained Planning for Multi-Agent Systems with Shared Resources. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 9 pages.

1 INTRODUCTION

Markov Decision Processes (MDPs) are a common model for *single-agent* planning and decision making under uncertainty. 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. In this paper, we consider the problem of offline policy synthesis, where agents plan ahead of time what actions they will take in any given situation. The class of MDPs with constraints extends MDPs such that actions also incur a cost that corresponds to the consumption of a resource. In an MDP with a constraint, the agent still wants to maximise their future expected reward, but they also need to constrain their resource usage.

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. This occurs as a results 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. 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. This is a restrictive constraint, and in some settings may be impossible to achieve. 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 [3]. 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. Another common approach is planning with a chance-constraint [5, 15, 27], which limits the percent of cases which exceed some resource consumption limit L. In other words, a chance-constraint

ensures that everything *except* the δ-tail of the cost distribution is bounded by L. However this approach provides no understanding over how bad the worst-cases get because it ignores the distribution within the δ-tail. 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) [6]. 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. This allows for a formal analysis of the worst-cases of the cost distribution.

Multi-agent planning under uncertainty can be modelled with a Multi-Agent MDP (MMDP) which considers the joint states and actions of all the participating agents [7]. In such cases, the goal of policy synthesis is to maximise the sum of the expected rewards across all agents. This model can be extended to MMDPs with constraints, where all agents use a shared resource. 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 policy. Any of the methods described above can be directly applied to MMDP with constraints, but this approach scales poorly, as the state and action spaces of an MMDP are exponential in the number of agents. A large body of work aims to mitigate this problem by instead focusing on **weakly coupled** MMDPs with constraints [21] which reason over individual agent models separately and consider only the shared resource jointly. The problem of planning for reward in weakly coupled MMDPs with constraints has been considered for a variety of constraint formulations, including constraining the worst-case cost [1, 32], constraining the expected cost [31, 35], and a chance-constraint [12, 14]. See [13] for a taxonomy of multi-agent constrained planning problems along with current algorithms. To our knowledge, there is no work studying risk-constraints in multi-agent MDPs.

This paper addresses this gap by tackling the problem of maximising for joint reward while constraining the CVaR of joint cost in a multi-agent system. Our main contributions are:

(1) formally defining the risk-constrained, multi-agent planning problem; and
(2) presenting an algorithm that solves the risk-constrained, multi-agent planning problem by using the notion of *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. Policy updates are carried out using a modification of the single-agent risk-constrained planning problem from [6]. Our empirical work 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.

2 PRELIMINARIES

Multi-Agent MDPs. We consider n agents, where each agent $i \in [n] = \{1, \dots, n\}$ has their own independent finite-horizon MDP $\mathcal{M}_i := \langle S_i, A_i, T_i, R_i, C_i, h \rangle$, 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 have a reward function $R_i : S_i \times A_i \to \mathbb{R}$, which describes how much reward is accrued after an action, and a cost function $C_i : S_i \times A_i \to \mathbb{N}^+$, which describes how much resource is consumed after an action. All agents have knowledge of their own MDP and share the same global time horizon h. A policy $\pi_i : S_i \times [h] \to A_i$ with $\pi_i(s_i, t) = a$ means that agent i should take action a at state s_i and timestep t. We denote the probability of an event D under π as $P_\pi[D]$ and the expectation of a random variable Y under π as $E_\pi[Y]$. We 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. Similarly, we define the random variable C_{i,π_i} to describe the cumulative cost agent i incurs when executing policy π_i over the entire time horizon h.

The joint *weakly-coupled* MMDP over all n agents is 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 $s = (s_1, \dots, s_n)$, $a = (a_1, \dots, a_n)$ and $s' = (s'_1, \dots, s'_n)$. 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 $C(s, a) = \sum_i C_i(s_i, a_i)$. h is the shared time horizon. A policy π for a weakly-coupled MMDP is defined by $\pi = \{\pi_i\}_{i \in [n]}$ for a set of single-agent polices π_i.

We define the random variable $\mathcal{R}_\pi$ to describe the cumulative reward from the MMDP when executing policy π over the entire time horizon h. Similarly, we define the random variable C_π to describe the cumulative cost from the MMDP when executing policy π over the entire time horizon h.

Conditional Value-at-Risk. CVaR originally emerged as a way of analysing the tails of a given probability distribution in finance [4, 26]. Given a random variable Z with cumulative distribution function $F(z)$, the Value-at-Risk (VaR) of Z for a given confidence level $\delta \in (0, 1]$ is defined as:

$$\text{VaR}_\delta(Z) = \min\{z | F(z) > 1 - \delta\}. \tag{1}$$

VaR can be interpreted as the value at which the $1 - \delta$ quantile of a distribution begins. Then, the CVaR of Z for a given confidence level $\delta \in (0, 1]$ is defined as:

$$\text{CVaR}_\delta(Z) = \frac{1}{\delta} \int_{1-\delta}^{1} \text{VaR}_{1-\gamma}(Z) d\gamma, \tag{2}$$

or equivalently as:

$$\text{CVaR}_\delta(Z) = E[Z | Z \geq \text{VaR}_\delta(Z)]. \tag{3}$$

CVaR is then the expected value of Z in the cases at which Z exceeds the $1 - \delta$ quantile. In other words, CVaR represents the expectation of Z in the worst $\delta * 100\%$ of cases.

We now focus our attention on a specific type of random variable, $Z = \sum_{i=1}^{n} Z_i$, where each Z_i is an independent random variable. In this context, it can be useful to define the risk contribution of each random variable Z_i to $\text{CVaR}_\delta(Z)$ [30, 33]. The risk contribution (RC) of variable Z_i is defined as:

$$\text{RC}_{\delta,Z}(Z_i) = E[Z_i | Z \geq \text{VaR}_\delta(Z)]. \tag{4}$$

Algorithm 1 iRMDP [6]

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 $\delta \in (0, 1]$

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

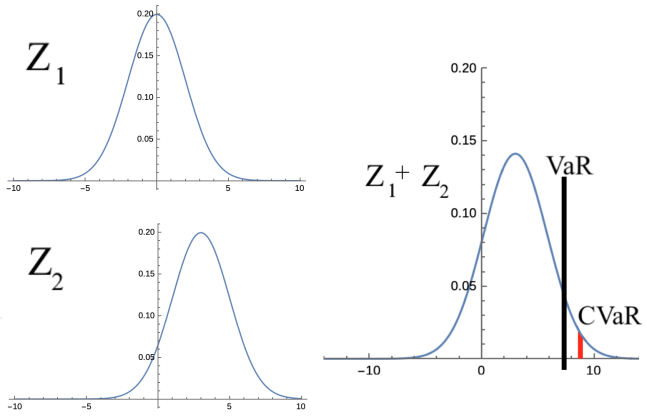

Figure 1: Two random variables, $Z_1 \sim \mathcal{N}(\mu = 0, \sigma^2 = 2)$ and $Z_2 \sim \mathcal{N}(\mu = 3, \sigma^2 = 2)$, along with their sum $Z \sim Z_1 + Z_2$. Included is the $\text{VaR}_{0.05}(Z)$ and $\text{CVaR}_{0.05}(Z)$ of Z.

Intuitively, 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.:

$$\text{CVaR}_\delta(Z) = \sum_{i=1}^{n} RC_{\delta,Z}(Z_i). \tag{5}$$

Example 1. *In Figure 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 cumulative value within the δ-tail of Z. CVaR = $E[Z|Z \geq VaR_\delta(Z)] = 8.8$ corresponds to the average cumulative value 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.

In this paper, the random variables under consideration are C_{i,π_i} for $i \in [n]$, i.e., the cost distribution that results from agent i executing policy π_i. Recall that $C_\pi = \sum C_{i,\pi_i}$ represents the joint cost distribution over all agents. $\text{RC}_{\delta,C_\pi}(C_{i,\pi_i})$ can then be interpreted as the risk that agent i contributes to the system.

iRMDP. iRMDP is an algorithm introduced in [6] which solves the single-agent risk-constrained planning problem. Given an MDP $\mathcal{M}$, a confidence level $\delta \in (0, 1]$, and a risk-bound $L \in \mathbb{N}^+$ over the resource constraint, the goal of the risk-constrained MDP is to find a policy $\pi^\star$ that maximises the cumulative reward, subject to the risk constraint:

$$\pi^\star = \arg\max_\pi \quad E_\pi[\mathcal{R}_\pi], \tag{6}$$

$$\text{s.t.} \quad \text{CVaR}_\delta[C_\pi] < L. \tag{7}$$

Because our single-agent policy update algorithm in Section 3.3.1 modifies iRMDP, we include for the reader a summary of the method. Note that, unlike [6], we describe iRMDP specifically for discrete state-space MDPs. The iRMDP algorithm solves the single-agent risk-constrained planning problem using a Lagrangian relaxation:

$$\min_{\lambda \geq 0} \max_\pi E_\pi[\mathcal{R}_\pi] + \lambda[L - \text{CVaR}_\delta[C_\pi]]. \tag{8}$$

The algorithm to optimise for Equation 8 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:

$$J_t(s,y) = \max_{\pi} \quad E_\pi \left[\sum_{\tilde{t}=t}^{h} R_\pi(s, c, \tilde{t}) \right] + \lambda^w \left[L - \text{CVaR}_\delta \left[y + \sum_{\tilde{t}=t}^{h} C_\pi(s, c, \tilde{t}) \right] \right]. \quad (9)$$

$J_t(s, y)$ is the sum of the future payoffs received by Equation 8 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 along with policy evaluation on $P_\pi[C_\pi \geq \beta^{w,v}]$ and $E_\pi[C_\pi | C_\pi \geq \beta^{w,v}]$. 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 π_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 starting in state s. Then, $V_0(s_0, 0) = P_\pi[C_\pi \geq \beta^{w,v}]$. 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 starting in state s. Then $Q_0(s_0, 0) = E_\pi[C_\pi | C_\pi \geq \beta^{w,v}]$.

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

3 RCA

3.1 Problem Description

A Risk-Constrained MMDP (RCMMDP) consists of a weakly-coupled MMDP $\mathcal{M}$, a confidence level $\delta \in (0, 1]$, and a risk-bound $L \in \mathbb{N}^+$ over the resource constraint. 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:

$$\pi^\star = \underset{\pi := \{\pi_i\}_{i \in [n]}}{\arg\max} \quad E_\pi \left[\sum_{i \in [n]} \mathcal{R}_{i, \pi_i} \right], \quad (10)$$

$$\text{s.t.} \quad \text{CVaR}_\delta \left[\sum_{i \in [n]} C_{i, \pi_i} \right] < L. \quad (11)$$

Example 2. *Consider the problem of running an advertising campaign over time through 1000 automated agents, as described in [8]. 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.

3.2 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.

3.3 A Risk Contribution Approach

To solve the RCMMDP problem defined in Equations (10-11), we present the Risk Contribution Approach (RCA) algorithm. RCA is an iterative algorithm that starts with a risk-neutral set of single-agent policies, 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 (11) 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: π_i = RiskNeutralPolicy($\mathcal{M}_i$)
3: **end for**
4: $var_0, cvar_0, \{rc_{0,i}\}_{i\in[n]}$ = CalcRisk($\mathcal{M}, \{\pi_i\}_{i\in[n]}, \delta$)
5: **for** u=1,2,...,until $cvar_{u-1} < L$ **do**
6: $j = \arg\max\{\frac{rc_{u-1,i}}{E[\mathcal{R}_{i,\pi_i}]} | i \in [n]\}$
7: $r\tilde{c}_j = rc_{u-1,j} - \gamma$
8: $\tilde{\beta} = var_{u-1} - \sum_{i\neq j} rc_{u-1,i}$
9: π_j = RCConstrainedPolicy($\mathcal{M}_j, r\tilde{c}_j, \tilde{\beta}$)
10: $var_u, cvar_u, \{rc_{u,i}\}_{i\in[n]}$ = 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. Line 4 calculates the risk associated with the initial policies using Monte Carlo trials. This yields the VaR of the *joint* cost, $VaR_\delta(\sum_{i\in[n]} C_{i,\pi_i})$, the CVaR of the *joint* cost, $CVaR_\delta(\sum_{i\in[n]} C_{i,\pi_i})$, 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 4.

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 tradeoff, 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 $r\tilde{c}_j := rc_{u-1,j} - \gamma$ (line 7).

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

$$E_\pi\left[C_{j,\pi_j} | \sum_{i\in[n]} C_{i,\pi_i} \geq VaR_\delta(\sum_{i\in[n]} C_{i,\pi_i})\right] \leq r\tilde{c}_j. \tag{12}$$

Exactly optimising for Equation 12 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 ($\text{VaR}_\delta(\sum_{i\in[n]} C_{i,\pi_i})$) with the VaR from the previous iteration's set of policies (denoted by var_{u-1}). This yields:

$$E_\pi\left[C_{j,\pi_j} | \sum_{i\in[n]} C_{i,\pi_i} \geq var_{u-1}\right] \leq r\tilde{c}_j. \tag{13}$$

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 12). $\text{VaR}_\delta(\sum_{i\in[n]} C_{i,\pi_i})$ will be similar to $\text{VaR}_\delta(C_{j,\pi_j^\star} + \sum_{i\neq j} C_{i,\pi_i})$ 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:

$$E_\pi[C_{j,\pi_j} | \sum_{i\in[n]} C_{i,\pi_i} \geq var_{u-1}] \tag{14}$$

$$= E_\pi[C_{j,\pi_j} | C_{j,\pi_j} + \sum_{i\neq j} C_{i,\pi_i} \geq var_{u-1}] \tag{15}$$

$$\approx E_\pi[C_{j,\pi_j} | C_{j,\pi_j} + \sum_{i\neq j} E_{\pi_i}[C_{i,\pi_i} | \sum_{i\neq j} C_{i,\pi_i} \geq var_{u-1}] \geq var_{u-1}] \tag{16}$$

$$= E_{\pi_j}[C_{j,\pi_j} | C_{j,\pi_j} \geq var_{u-1} - \sum_{i\neq j} E_{\pi_i}[C_{i,\pi_i} | \sum_{i\neq j} C_{i,\pi_i} \geq var_{u-1}]] \tag{17}$$

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

$$E_{\pi_j}\left[C_{j,\pi_j} | C_{j,\pi_j} \geq var_{u-1} - \sum_{i\neq j} rc_{u-1,i}\right] \leq r\tilde{c}_j. \tag{18}$$

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 18) with RCConstrainedPolicy, described in Section 3.3.1, which is a modification of the single-agent iRMDP. Finally, line 10 calculates the risk associated with the current policies in the same method as line 4. This process is repeated until the risk constraint is met, at which point the current policies are returned.

3.3.1 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:

$$\pi^\star = \arg\max_\pi \quad E_{\pi_j}\left[\mathcal{R}_{j,\pi_j}\right], \tag{19}$$

$$\text{s.t.} \quad E_{\pi_j}\left[C_{j,\pi_j} | C_{j,\pi_j} \geq \tilde{\beta}\right] \leq r\tilde{c}_j. \tag{20}$$

Algorithm 3 accomplishes this by modifying iRMDP, 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 $\tilde{\beta}$.

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

Like iRMDP we solve the Lagrangian relaxation:

$$\min_{\lambda \geq 0} \max_{\pi_j} E_{\pi_j}\left[\mathcal{R}_{j,\pi_j}\right] + \lambda\left[r\tilde{c}_j - E_{\pi_j}\left[C_{j,\pi_j} | C_{j,\pi_j} \geq \tilde{\beta}\right]\right], \quad (21)$$

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. But 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.

4 EVALUATION

We evaluated the performance of our multi-agent algorithm against iRMDP on two benchmark domains: Maze from [32] and advertising budgets from [8].

4.1 Implementation Details

For our baseline, we use **iRDMP** as described in Algorithm 2. 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. We also compare to a Risk Neutral algorithm, label as **RN**, which only optimises for reward. Policy synthesis for this method is done with the PRISM model checker [18]. 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 $E[C_{i,\pi_i}]$ 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 for both algorithms is set proportionally to C. The iRMDP details are the same as for the baseline usage. 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.

4.2 The Maze Domain

4.2.1 Domain Description. We first modified the Maze domain from [32] to use a multi-unit resource. 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. 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 [19]. Once an agent is in a task location, they can choose to perform a task action, at which point their execution

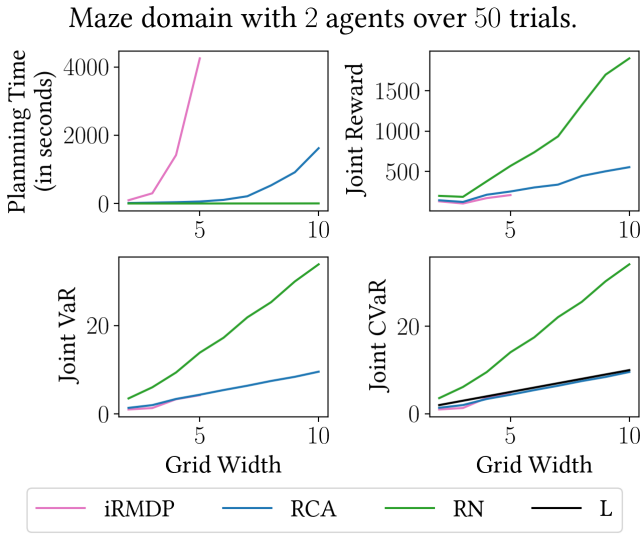

Figure 2: An analysis of RCA and iRMDP on increasing large instances of the Maze domain for 2 agents. Each data point corresponds 50 trials.

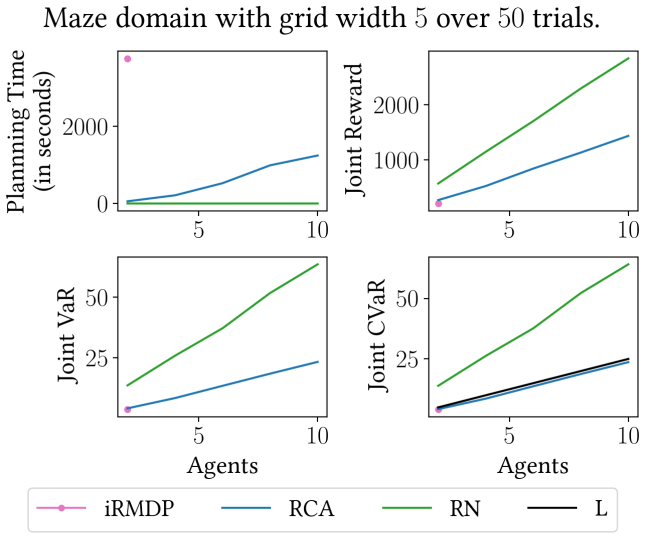

Figure 3: An analysis of RCA and iRMDP on increasing numbers of agents in the Maze domain of gridsize 5 by 5. Each data point corresponds 50 trials.

ends. Agents' only interaction with each other comes in the form of a global resource constraint. 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. This limit was chosen to strike a balance between being higher and thus effectively unconstrained and lower and thus too restrictive for interesting action choices. 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.

4.2.2 Results. As expected, iRMDP scales poorly with respect to planning time, as seen in both Figure 2, which varies the size of the single-agent state space, and Figure 3, 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 3 this is only a 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. The CVaR is constrained by L as expected for iRMDP and RCA methods, in both Figures 2 and 3. In both Figures, RN outperforms both iRMDP and RCA in terms of time and joint reward, but only because it ignores the CVaR constraint.

4.3 The Advertising Domain

4.3.1 Domain Description. Recall the advertising budget allocation domain in Example 2, originally from [8], with 1000 agents, 15 states, and 5 actions per state. The advertiser is rewarded only when an agent purchases their product, and pays a monetary cost that is action dependent. This MDP contains only one configuration (as in [8]), and as such results are over a single execution of RCA. All

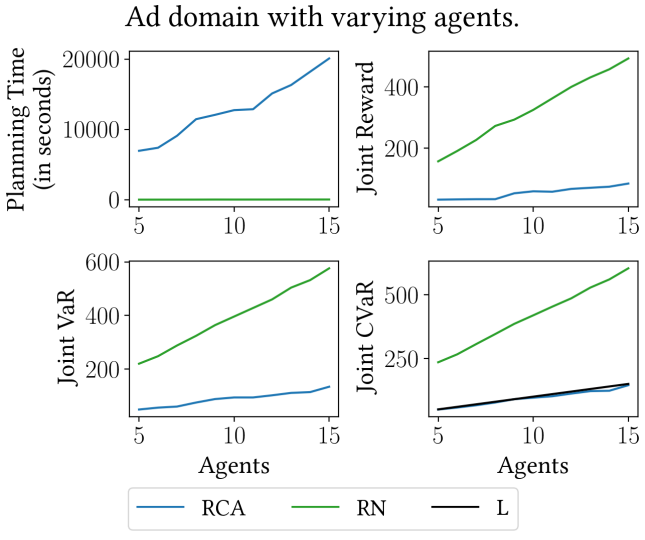

Figure 4: Algorithm performance on the Advertising domain with increasing numbers of agents.

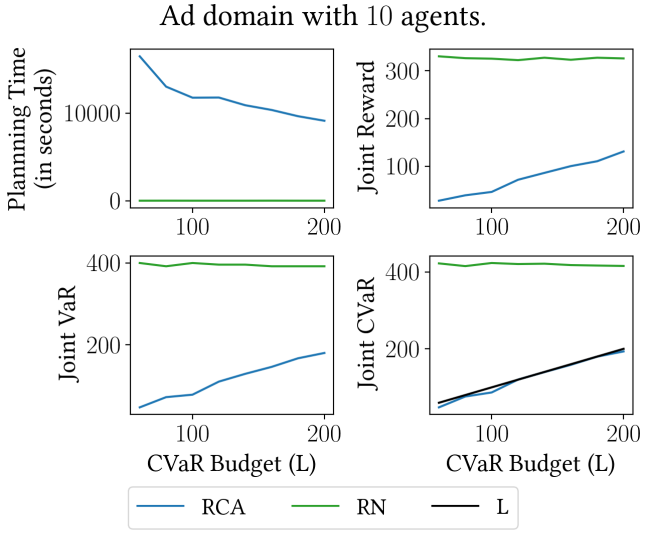

Figure 5: Algorithm performance on the Advertising domain with increasing budgets.

agents' MDPs are identical but independent, so the advertiser can pursue different strategies for different agents. We slightly modify the domain by setting a time horizon of $h = 30$ to make the risk-constrained problem feasible to solve, and we modify the objective to undiscounted reward to match our setting. In all experiments, $\delta = 0.05$. In Figure 4 the joint CVaR is limited by $L = 10n$. In Figure 5, L is varied along the x-axis.

4.3.2 Results. As in the Maze domain, in Figure 4 the planning time increases with the number of agents (and the increased CVaR requirements). The CVaR is successfully constrained through all numbers of agents in Figure 4. In Figure 5, 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 is successfully constrained through all numbers of budgets in Figure 5. 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.

5 RELATED WORK

There is a wealth of literature that studies the risk associated with sequential decision making under uncertainty, and in particular risk through the lens 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. Though our goal is to achieve (4) in a multi-agent setting, we include other categories for completeness. 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 [11, 20, 36] and reinforcement learning [17, 23, 25, 28, 29]. These approaches differ from our approach (and the previous MDP with constraints literature) in that the primary goal is to manage some cost, instead of optimising for reward while managing a cost. 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 [9, 24] and reinforcement learning [16, 34]. 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. In category (3), maximising the CVaR of a reward function and constraining the CVaR of a cost function, [2] does consider both a reward function and a cost function, but consider the CVaR of both the reward and cost constraint, which allows for similar solution methods to category (2). 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. Recently, [6] devised an algorithm to solve the RCMDP planning problem for a single-agent. The methodology of this approach is discussed in Section 2. One could extend their method to MMDPs as we describe in Section 3.2, but as we show in our evaluation, this scales poorly. [10] and [22] solve a similar problem with single-agent reinforcement learning. To our knowledge, our work is the first to study the risk-constrained MMDP (RCMMDP) problem in classical planning or multi-agent reinforcement learning.

6 CONCLUSIONS AND FUTURE WORK

Constrained planning problems in multi-agent systems under uncertainty require unique solutions to handle the state space explosion that arises from MMDPs. In this paper, we tackled the Risk-Constrained MMDP problem, i.e., the problem of optimising for expected reward while constraining the joint CVaR of a shared resource. 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. With this method, we avoid the the state and action space explosion of solving a joint model by instead iteratively updating only one agent's policy. We evaluated RCA against a single-agent risk-constrained solver iRMDP on two benchmarks, the Maze domain from [32] and an advertising domain from [7]. We demonstrate that not only can RCA successfully solve the RCMMDP problem, but also it significantly outperforms iRMDP in terms of planning time. Future work includes warm-starting successive calls to the RCConstrainedPolicy and evaluating RCA on real-world data. Additionally, we hope to expand our approach to constraining other *coherent risk measures*.

ACKNOWLEDGMENTS

This work was supported by a gift from Amazon Web Services. Gautier was supported by the AIMS Centre for Doctoral Training. Lacerda and Hawes were supported by the EPSRC Programme Grant 'From Sensing to Collaboration' [EP/V000748/1]. Wooldridge was supported by a UKRI AI World Leading Researcher Fellowship [EP/W002949/1].

REFERENCES

[1] Pritee Agrawal, Pradeep Varakantham, and William Yeoh. 2016. Scalable Greedy Algorithms for Task/Resource Constrained Multi-Agent Stochastic Planning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*.

[2] Mohamadreza Ahmadi, Ugo Rosolia, Michel D. Ingham, Richard M. Murray, and Aaron D. Ames. 2021. Constrained Risk-Averse Markov Decision Processes. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence*.

[3] Eitan Altman. 1999. *Constrained Markov Decision Processes: Stochastic Modeling*. Routledge.

[4] Philippe Artzner, Freddy Delbaen, Jean-Marc Eber, and David Heath. 1999. Coherent Measures of Risk. *Mathematical Finance* (1999).

[5] Benjamin J Ayton and Brian C Williams. 2018. Vulcan: a Monte Carlo Algorithm for Large Chance Constrained MDPs with Risk Bounding Functions. In *Computing Research Repository*.

[6] Vivek Borkar and Rahul Jain. 2014. Risk-Constrained Markov Decision Processes. *Transactions on Automatic Control* 59, 9 (2014), 2574–2579.

[7] Craig Boutilier. 1996. Planning, Learning and Coordination in Multiagent Decision Processes. In *Proceedings of the Sixth Conference on Theoretical Aspects of Rationality and Knowledge*.

[8] Craig Boutilier and Tyler Lu. 2016. Budget Allocation using Weakly Coupled, Constrained Markov Decision Processes. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*.

[9] Yinlam Chow and Mohammad Ghavamzadeh. 2014. Algorithms for CVaR optimization in MDPs. *Advances in neural information processing systems* 27 (2014).

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

[11] Yinlam Chow, Aviv Tamar, Shie Mannor, and Marco Pavone. 2015. Risk-Sensitive and Robust Decision-Making: a CVaR Optimization Approach. In *In Proceedings of the Twenty-Eighth International Conference on Neural Information Processing Systems*.

[12] Frits de Nijs, Erwin Walraven, Mathijs de Weerdt, and Matthijs Spaan. 2017. Bounding the Probability of Resource Constraint Violations in Multi-Agent MDPs. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*.

[13] Frits de Nijs, Erwin Walraven, Mathijs De Weerdt, and Matthijs Spaan. 2021. Constrained Multiagent Markov Decision Processes: A Taxonomy of Problems and Algorithms. *Journal of Artificial Intelligence Research* 70 (2021), 955–1001.

[14] Anna Gautier, Bruno Lacerda, Nick Hawes, and Michael Wooldridge. 2023. Multi-Unit Auctions for Allocating Chance-Constrained Resources. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence*.

[15] William B. Haskell and Rahul Jain. 2015. A Convex Analytic Approach to Risk-Aware Markov Decision Processes. *SIAM Journal on Control and Optimization* 53, 3 (2015), 1569–1598.

[16] Takuya Hiraoka, Takahisa Imagawa, Tatsuya Mori, Takashi Onishi, and Yoshimasa Tsuruoka. 2019. Learning Robust Options by Conditional Value at Risk Optimization. In *Proceedings of the Thirty-Third International Conference on Neural Information Processing Systems*.

[17] Ramtin Keramati, Christoph Dann, Alex Tamkin, and Emma Brunskill. 2020. Being Optimistic to Be Conservative: Quickly Learning a CVaR Policy. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*.

[18] Marta Kwiatkowska, Gethin Norman, and David Parker. 2002. PRISM: Probabilistic Symbolic Model Checker. In *In Proceedings of the Twelfth International Conference on Computer Performance Evaluation, Modelling Techniques and Tools*.

[19] Morteza Lahijanian, Maria Svorenova, Akshay A Morye, Brian Yeomans, Dushyant Rao, Ingmar Posner, Paul Newman, Hadas Kress-Gazit, and Marta Kwiatkowska. 2018. Resource-Performance Tradeoff Analysis for Mobile Robots. *Robotics and Automation Letters* 3, 3 (2018), 1840–1847.

[20] Xiaocheng Li, Huaiyang Zhong, and Margaret L Brandeau. 2021. Quantile Markov Decision Processes. *Operations Research* 70, 3 (2021), 1428–1447.

[21] Nicolas Meuleau, Milos Hauskrecht, Kee-Eung Kim, Leonid Peshkin, Leslie Pack Kaelbling, Thomas L Dean, and Craig Boutilier. 1998. Solving Very Large Weakly Coupled Markov Decision Processes. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*.

[22] LA Prashanth and Michael C Fu. 2022. Risk-Sensitive Reinforcement Learning via Policy Gradient Search. *Foundations and Trends in Machine Learning* 15, 5 (2022), 537–693.

[23] Wei Qiu, Xinrun Wang, Runsheng Yu, Rundong Wang, Xu He, Bo An, Svetlana Obraztsova, and Zinovi Rabinovich. 2021. RMIX: Learning Risk-Sensitive Policies for Cooperative Reinforcement Learning Agents. In *Proceedings of the Thirty-Fourth International Conference on Neural Information Processing Systems*.

[24] Marc Rigter, Paul Duckworth, Bruno Lacerda, and Nick Hawes. 2022. Planning for Risk-Aversion and Expected Value in MDPs. In *Proceedings of the Thirty-Second International Conference on Automated Planning and Scheduling*.

[25] Marc Rigter, Bruno Lacerda, and Nick Hawes. 2021. Risk-Averse Bayes-Adaptive Reinforcement Learning. In *Proceedings of the Thirty-Fourth International Conference on Neural Information Processing Systems*.

[26] R Tyrrell Rockafellar and Stanislav Uryasev. 2000. Optimization of Conditional Value-at-Risk. *Journal of Risk* 2 (2000), 21–42.

[27] Pedro Santana, Sylvie Thiébaux, and Brian Williams. 2016. RAO*: An Algorithm for Chance-Constrained POMDP's. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*.

[28] Aviv Tamar, Yinlam Chow, Mohammad Ghavamzadeh, and Shie Mannor. 2015. Policy Gradient for Coherent Risk Measures. In *Proceedings of the Twenty-Eighth International Conference on Neural Information Processing Systems*.

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

[30] Dirk Tasche. 1999. Risk Contributions and Performance Measurement. (1999).

[31] Erwin Walraven and Matthijs TJ Spaan. 2018. Column Generation Algorithms for Constrained POMDPs. *Journal of Artificial Intelligence Research* 62 (2018), 489–533.

[32] Jianhui Wu and Edmund H Durfee. 2010. Resource-Driven Mission-Phasing Techniques for Constrained Agents in Stochastic Environments. *Journal of Artificial Intelligence Research* 38 (2010), 415–473.

[33] Yasuhiro Yamai and Toshinao Yoshiba. 2002. Comparative Analyses of Expected Shortfall and Value-at-Risk: Their Estimation Error, Decomposition, and Optimization. *Monetary and Economic Studies* 20, 1 (2002), 87–121.

[34] Chengyang Ying, Xinning Zhou, Hang Su, Dong Yan, Ning Chen, and Jun Zhu. 2022. Towards Safe Reinforcement Learning via Constraining Conditional Value-at-Risk. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*.

[35] Kirk A Yost and Alan R Washburn. 2000. The LP/POMDP marriage: Optimization with imperfect information. *Naval Research Logistics* 47, 8 (2000), 607–619.

[36] Pengqian Yu, William B. Haskell, and Huan Xu. 2018. Approximate Value Iteration for Risk-Aware Markov Decision Processes. *Transactions on Automatic Control* 63, 9 (2018), 3135–3142.