Multi-Unit Auctions for Allocating Chance-Constrained Resources

Anna Gautier, Bruno Lacerda, Nick Hawes, Michael Wooldridge
University of Oxford
Oxford, United Kingdom
anna.gautier@eng.ox.ac.uk, bruno@robots.ox.ac.uk, nickh@robots.ox.ac.uk, mjw@cs.ox.ac.uk

Abstract

Sharing scarce resources is a key challenge in multi-agent interaction, especially when individual agents are uncertain about their future consumption. We present a new auction mechanism for preallocating multi-unit resources among agents, while limiting the chance of resource violations. By planning for a chance constraint, we strike a balance between worst-case approaches, which under-utilise resources, and expected-case approaches, which lack formal guarantees. We also present an algorithm that allows agents to generate bids via multi-objective reasoning, which are then submitted to the auction. We then discuss how the auction can be extended to non-cooperative scenarios. Finally, we demonstrate empirically that our auction outperforms state-of-the-art techniques for chance-constrained multi-agent resource allocation in complex settings with up to hundreds of agents.

Introduction

When multiple independent agents are operating in the same environment, it often becomes necessary for them to share limited resources. 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 our work, we focus on *multi-unit resources* like battery power and bandwidth. Our goal is to design a system that preallocates the resources among the agents. It is important for a resource distribution system in this context to consider that agents are working in an uncertain environment. In this paper, 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 usage is resolved, 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 is known as a Chance Constrained Multi-Agent MDP, and was originally introduced in de Nijs et al. (2017) .

In this paper, 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. A Mixed Integer Linear Program (MILP) *allocates resources* to agents while ensuring a chance constraint over the probability of exceeding the resource limit is met over all agents. We then consider how multi-objective reasoning can be used to *generate single-agent bids*. We also *present a pricing structure* that can be used to apply the mechanism to non-cooperative settings. Finally, we *empirically evaluate ACCR* against state-of-the-art approaches using Maze (Wu and Durfee 2010), a classic multi-agent constrained resource domain, and an advertising budget allocation domain from Boutilier and Lu (2016). Our results in both domains outperform the state of the art for up to hundreds of agents.

Related Work

In this section we briefly summarise the literature on pre-planning over single and multi-agent MDPs with **budget-constrained** resources, or resources that are constrained in summation over a time horizon (de Nijs et al. 2021). Altman (1999) introduces the concept of a Constrained MDP (CMDP), which consists of an MDP with a resource limit L. They also introduce a Linear Program (LP) which makes sure that the expected cost of the agent's policy is less than L. Chance constraints bound the probability that the cost of the agent's policy violated the resource limit, instead of planning for the expected costs. Chance-Constrained MDPs (CCMDPs) allow for constraint violations in unlikely scenarios, but still guarantee that the constraint is met with a specified likelihood. Haskell and Jain (2015) introduce an LP solution method for policy generation in CCMDPs that utilises a convex analytic method. Santana, Thiébaux, and Williams (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.

Multi-agent MDPs (MMDPs) extend these concepts to a multi-agent system. Many single-agent methods can be trivially extended to MMDPs. Solving CMDPs with an expected resource limit can be trivially extended to the multi-agent domain, i.e. Constrained MMDPs (CMMDPs), by considering a joint model as in (Boutilier 1996). This approach allows a wide range of methods to be applicable to MMDPs, but does not scale because the policy search space is 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** CMMDP (Meuleau et al. 1998). For example, Column Generation solves the weakly-coupled 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). 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. Agrawal, Varakantham, and Yeoh (2016) present an approximate worst-case solution problem, which uses a decentralised greedy algorithm to allocate resources. 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. Our work also solves the CCMMDP problem, but does so without making a conservative approximation. For an in-depth overview of multi-agent resource allocation techniques, including techniques designed for other types of resources and online resource allocation, see de Nijs et al. (2021).

Auctioning approaches have been widely proposed in robotics, particularly for coordination of teams of robots. Early work on this topic uses combinatorial auctions (Hunsberger and Grosz 2000) and first-price one-round auctions (Gerkey and Mataric 2002) to distribute tasks across a set of agents. Later, Lagoudakis et al. (2005) propose a sequential single item (SSI) auction mechanism for multi-robot routing and task allocation. This approach combines the advantages of parallel single item auctions and combinatorial auctions, achieving high quality task allocations with low computational effort (Koenig et al. 2006). There has also been work on using auctions to distribute parts of a global task across a team of robots, in the context of planning under uncertainty (Capitan et al. 2013; Schillinger, Bürger, and Dimarogonas 2018). Unlike these methods, we auction shared resources. Amir, Sharon, and Stern (2015) and Gautier et al. (2022) apply combinatorial auctions to multi-agent path planning, which effectively auctions space as a resource. Dolgov and Durfee (2006) use a Generalised Vickrey Auction to allocate resources in a MMDP with a worst-case constraint. Our method differs in that it can consider uncertainty over resource use, through a chance-constraint.

Preliminaries

Chance Constrained Multi-Agent MDPs. We consider n agents, where each agent $i \in [n] = \{1, \ldots, n\}$ has their own finite-horizon MDP $\mathscr{M}_i := \langle S_i, A_i, T_i, R_i, C_i, h \rangle$, 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. All agents share a 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 a policy π as $P_\pi[D]$ and the expectation of a random variable Y under the policy π as $E_\pi[Y]$. We define the random variable $\mathscr{R}_{i,\pi_i}$ 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 $\mathscr{C}_{i,\pi_i}$ to describe the cumulative cost agent i incurs when executing policy π_i over the entire time horizon h.

The joint MMDP over all n agents is represented by $\mathscr{M} := \langle S, A, T, R, C, h \rangle$ 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 $C(s, a) = \sum_i C_i(s_i, a_i)$. 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.

A Chance-Constrained Multi-agent MDP (CCMMDP) is a triple $\langle \mathscr{M}, L, \delta \rangle$. $\mathscr{M}$ denotes the MMDP. $L \in \mathbb{N}^+$ 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.,

$$\sum_{i\in[n]} \sum_{t\in[h]} C_i(s_{i,t}, a_{i,t}) > L. \tag{1}$$

This type of constraint is sometimes referred to as a budget constraint (de Nijs et al. 2021). Finally, $\delta \in [0,1)$ denotes the limit on the probability of constraint violation. 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:

$$\pi^\star = \underset{\pi := \{\pi_i\}_{i\in[n]}}{\arg\max} \quad E_\pi\left[\sum_{i\in[n]} \mathscr{R}_{i,\pi_i}\right], \tag{2}$$

$$\text{s.t.} \quad P_\pi\left[\sum_{i\in[n]} \mathscr{C}_{i,\pi_i} > L\right] < \delta. \tag{3}$$

Multi-Unit Auctions. A **multi-unit auction** is a set of rules that determines how to distribute L identical resources to n 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 and $b_{i,k}$ is the amount that agent i would be willing to pay to receive k resources. The auctioneer's task is to allocate the resources as efficiently as possible. This can be expressed as a Mixed-Integer Linear Program (MILP):

$$\text{maximise} \quad \sum_{i\in[n]} \sum_{k\in[L]} b_{i,k} x_{i,k} \tag{4}$$

$$\text{subject to} \quad \sum_{i\in[n]} \sum_{k\in[L]} k x_{i,k} \leq L \tag{5}$$

$$\sum_{k\in[L]} x_{i,k} \leq 1 \quad \forall i \in [n] \tag{6}$$

$$x_{i,k} \in \{0,1\} \quad \forall k \in [L], i \in [n] \tag{7}$$

Decision variables $x_{i,k}$ correspond to agent i being allocated k resources and are integral. The auctioneer directly optimises for the sum of bids (4) while constraining the total number of resources (5) and ensuring that all agents are allocated a single bid (6, 7). This is referred to as the Winner Determination Problem (Lehmann, Müller, and Sandholm 2006), and results in an allocation $\mathscr{F}^\star = \{(k_1^\star, b_1^\star), \dots, (k_n^\star, b_n^\star)\}$. Note that this model does not consider uncertainty. We introduce a new auction structure that allows for it next.

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 above. 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.

The ACCR Protocol. In ACCR, each agent submits a list of tuples of the form $B_{i,\alpha} = (k_{i,\alpha}, b_{i,\alpha}, \varepsilon_{i,\alpha})$. 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 $\varepsilon_{i,\alpha}$, 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 $\pi_{i,\alpha}$ and $\varepsilon_{i,\alpha}$ represents the probability that the total number of resources consumed while executing $\pi_{i,\alpha}$ exceeds $k_{i,\alpha}$. 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 Mixed Integer Nonlinear Program (MINLP).

$$\text{maximise} \quad \sum_{i\in[n]} \sum_{\alpha\in[m_i]} b_{i,\alpha} x_{i,\alpha} \tag{8}$$

$$\text{subject to} \quad \sum_{i\in[n]} \sum_{\alpha\in[m_i]} k_{i,\alpha} x_{i,\alpha} \leq L \tag{9}$$

$$\prod_{i\in[n]} \left(1 - \sum_{\alpha\in[m_i]} \varepsilon_{i,\alpha} x_{i,\alpha}\right) \geq 1 - \delta \tag{10}$$

$$\sum_{\alpha\in[m_i]} x_{i,\alpha} \leq 1 \quad \forall i \in [n] \tag{11}$$

$$x_{i,\alpha} \in \{0,1\} \quad \forall \alpha \in [m_i], i \in [n] \tag{12}$$

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 (8) while constraining the total number of resources (9) and ensuring that all agents are allocated a single bid (11, 12). 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. Satisfying (10) is equivalent to ensuring that the chance constraint in (3) holds. To see this, note that because the joint declared resource use of all agents is less than

L, equation (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 (10). Constraint (10) is nonlinear, which makes the problem a MINLP. However, it can be translated into a linear constraint.

Proposition 1. *Substituting Constraint (10) in the above MINLP with the following Constraint (13) will result in a MILP with an equivalent solution:*

$$\sum_{i \in [n]} \sum_{\alpha \in [m_i]} x_{i,\alpha} \left(\log \left(1 - \varepsilon_{i,\alpha} \right) \right) \geq \log \left(1 - \delta \right). \tag{13}$$

Proof. First, note that Constraint (10) is equivalent to the following constraint, because $0 \leq \delta < 1$:

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

Next, we see that,

$$\log \left(\prod_{i \in [n]} \left(1 - \sum_{\alpha \in [m_i]} \varepsilon_{i,\alpha} x_{i,\alpha} \right) \right) \tag{14}$$

$$= \sum_{i \in [n]} \left(\log \left(1 - \sum_{\alpha \in [m_i]} \varepsilon_{i,\alpha} x_{i,\alpha} \right) \right) \tag{15}$$

$$= \sum_{i \in [n]} \sum_{\alpha \in [m_i]} x_{i,\alpha} \log \left(1 - \varepsilon_{i,\alpha} \right) \tag{16}$$

(15) and (16) are equivalent because by Constraint (12) and Constraint (11):

- $x_{i,\alpha} \in \{0,1\}$, 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 equivalence is discussed in depth in the Appendix.

Thus replacing Constraint (10) in our original MINLP with Constraint (13) yeilds an equivalent MILP that solves the chance-constrained allocation problem more efficiently than the original optimisation problem. □

We denote the allocation yielded by solving the MILP as $\mathscr{F}^\star = \{(k_1^\star, b_1^\star, \varepsilon_1^\star), \ldots, (k_n^\star, b_n^\star, \varepsilon_n^\star)\}$.

Single-Agent Decision Making

ACCR requires that each agent can generate bids of the form $B_{i,\alpha} = (k_{i,\alpha}, b_{i,\alpha}, \varepsilon_{i,\alpha})$. 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 $\mathscr{M}_i = \langle S_i, A_i, T_i, R_i, C_i, h \rangle$, agents first need to extend their MDP to a new MDP $\tilde{\mathscr{M}}_i := \langle \tilde{S}_i, A_i, \tilde{T}_i, \tilde{R}_i, \tilde{C}_i, h \rangle$. The new state space is defined by $\tilde{S}_i = S_i \times [h] \times \mathbb{N}^+$, which allows $\tilde{\mathscr{M}}_i$ 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 $\tilde{T}_i : \tilde{S}_i \times A_i \times \tilde{S}_i \to [0,1]$:

$$\tilde{T}_i(\tilde{s}, \tilde{a}, \tilde{s}') = \begin{cases} T_i(s,a,s') & \text{if } T_i(s,a,s') > 0, \\ & \tilde{s} = (s,t,c), \text{ and} \\ & \tilde{s}' = (s', t+1, c + C(s,a)) \\ 0 & \text{otherwise.} \end{cases} \tag{17}$$

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 increase their cumulative reward and that decrease their probability of exceeding k, represented by the following two properties respectively:

$$\max_{\pi_i} E_{\pi_i} \left[\mathscr{R}_{i,\pi_i} \right] \tag{18}$$

$$\min_{\pi_i} P_{\pi_i} \left[c_h \geq k \text{ for } \tilde{s}_h = (s_h, h, c_h) \right] \tag{19}$$

These objectives are conflicting, meaning that improving one objective may come at the expense of the other. As a result, we care about **Pareto optimal policies**, which are policies such that no other policy strictly improves one objective without worsening the others. The set of such policies is called the **Pareto frontier**. We propose that agent's generate their bids by choosing policies in the 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, Kwiatkowska, and Parker 2012). For each such 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 (18), and ε corresponds to the current Pareto point's value for (19).

After each agent generates a list of bids, they are sent to the auctioneer. We require that agents only send bids with $\varepsilon < 1$. This is required so that the auctioneer can compute $\log(1 - \varepsilon)$ in the MILP. This requirement is without loss of generality because bids with $\varepsilon = 1$ 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 $\varepsilon > \delta$ 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).

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 (2) constrained by Equation (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:

$$\pi_i^* = \arg\max_{\pi_i} E_{\pi_i}\left[\sum_{t\in[h]} \mathscr{R}_{i,\pi_i}\right] \tag{20}$$

Agents can maximise their reward by lying about how much they value resources during bidding. To prevent this strategy, non-cooperative auctions use prices to incentivise agents to tell the truth. The non-cooperative multi-unit auction 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 only lose value if they decide to over or under bid.

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. The prices are charged after each agent executes their private policies $\pi_i^\star$ associated to $(k_i^\star, b_i^\star, \varepsilon_i^\star)$, 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:

$$p_i = \begin{cases} 0 & \text{if } k_i^r \leq k_i^\star \\ \dfrac{1}{\varepsilon_i^\star} \cdot p_i^v & \text{otherwise.} \end{cases} \tag{21}$$

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. A formal analysis can be found in the Appendix.

Evaluation

To evaluate the performance of ACCR, we compared its performance to a series of baselines on the benchmark domain Maze (Wu and Durfee 2010), and an advertising budget allocation MDP from Boutilier and Lu (2016).

Methods

We compared ACCR to four other methods. The **Multi-agent Markov Decision Process** (MMDP) method solves the chance constraint optimally by planning over the joint MMDP. To solve the chance-constrained problem, we extend the MMDP described in the preliminaries to include cumulative cost, similar to the process used for single-agent bid generation. Then we generate a Pareto frontier 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 **Constrained Markov Decision Process** (CMDP) method is implemented as described in Altman (1999). This centralised approach considers individual agents' MDPs expanded to include the current timestep. The solution method consists of an LP which, like ACCR, optimises for expected reward. Unlike ACCR, the constraint corresponding to the resource limit only ensures that the limit is met in expectation. This means that the method ignores the chance-constraint and does not limit the probability of resource violations. As a result, this method provides an upper bound for the solution value of a chance-constrained algorithm like ACCR, but should not be seen as a direct comparison.

The chance-constrained **Column Generation** (CG) method combines two techniques. First, it uses column generation which solves the same problem as the CMDP but in a decentralised manner, as in (Yost and Washburn 2000). Like the CMDP 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 CMDP method. Because it is decentralised, it is much faster than CMDP. The second technique the chance-constrained CG uses is a Hoeffding bound, as in (de Nijs et al. 2017). L_h is an artificially lowered resource limit that is based on L, δ and the maximum resource usage of each agent. In the LP, L_h is used instead of L, so the algorithm returns policies limited by L_h in expectation. L_h is designed so this will also limit the chance constraint, i.e., if the policies uses less that L_h 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_h can be too conservative, de Nijs et al. (2017) suggest a method to dynamically relax L_h. After each run of CG, the **Dynamic Column Generation** (CG$_d$) method runs Monte Carlo trials to estimate the probability distribution over resources and modify L_h accordingly. This process is repeated until the chance constraint is met without slack.

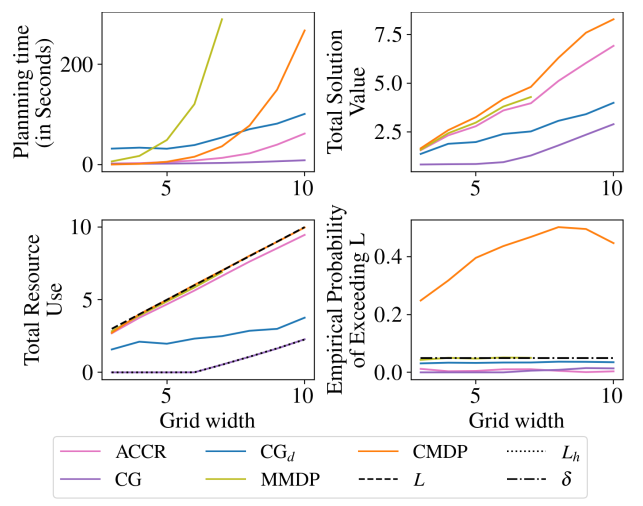

Figure 1: Algorithm performance on Maze with increasing state space. Trials are performed with 2 agents.

For all methods, LPs and MILPs 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, Norman, and Parker 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.

The Maze Domain

Domain Description. We first modified the Maze domain from (Wu and Durfee 2010) 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. Once an agent is in a task 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 $L = \frac{hn}{4}$, 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 $\delta = 0.05$. Each data point in Figure 1 and 2 represent the average over 50 trials. All methods timeout at 500 seconds.

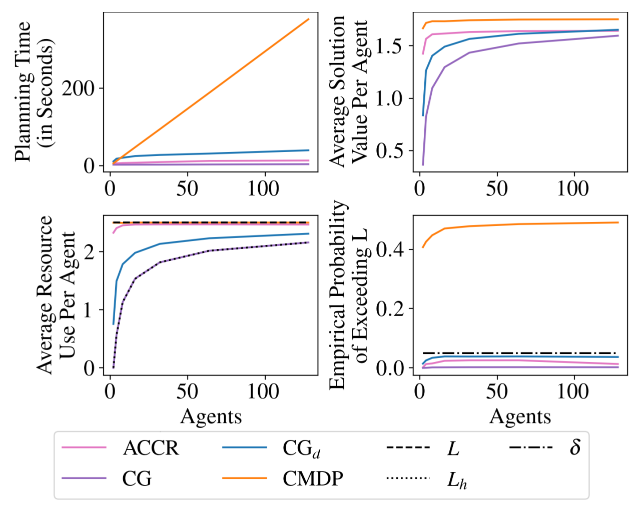

Figure 2: Algorithm performance on Maze with increasing agents. Trials were performed with grid width 5.

Results. In Figure 1, 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 CMDP 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_h to L. Because L_h 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_h 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_h. In ACCR, the auctioneer more flexibility to allocate all of the resources. Even as the empirical probability of exceeding the resource limit fluctuates, the solution value remains high. ACCR resource utilization is on par with the CMDP and MMDP approach. But like the CG approach, all planning over MDPs in ACCR is done on the single-agent model, allowing it compute a result more quickly than the joint approaches.

In Figure 2, we see how the decentralised natures of ACCR and CG allow for a significant speed up as the number of agents increases, compared to CMDP. 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_h approaches L as the number of agents increases, the both 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 number of agents.

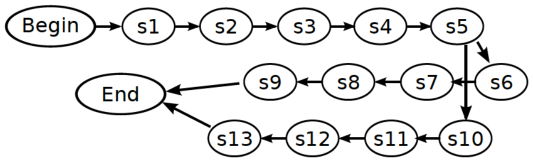

Figure 3: Synthetic Advertising Domain MDP from Boutilier and Lu (2016).

The Advertising Domain

Domain Description. In the Synthetic Advertising Domain from Boutilier and Lu (2016), an advertiser must choose a strategy to allocate a monetary budget among 1000 different agents to convert documented interest into sales. Each agent is described by a 15 state MDP which can be split into 3 parts: generic interest in the product category; interest in the advertiser's specific product: or, interest in a competitor's product. The advertiser has five advertising strategies at each state, which start at a zero-cost/no intervention strategy and increasingly become more costly, and more effective at moving the agents toward purchasing the advertiser's product. All actions are stochastic, having some probability of the agent exiting the process. When an agent purchases either the competitor's or advertiser's product, they also exit the process. The advertiser is rewarded when an agent purchases their product. The MDP is illustrated in Figure 3. All agents' MDPs are identical but independent, so the advertiser can pursue different strategies for different agents. 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.

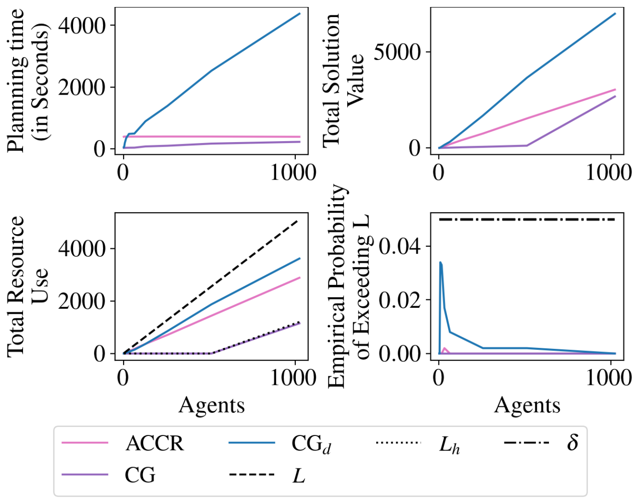

Figure 4: Algorithm performance on Synthetic Advertising Domain with increasing numbers of agents.

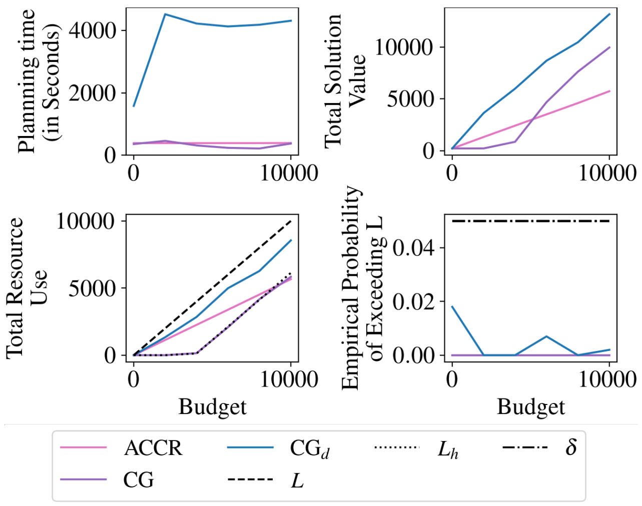

Figure 5: Algorithm performance on Synthetic Advertising Domain with increasing budgets.

Results. Figure 4 and Figure 5 show that on this domain, CG with dynamic relaxation performs the best, but is much more time consuming than the other algorithms. In Figure 4, 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 5, 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.

Conclusions

We have presented ACCR, a new auction-based mechanism for allocating chance-constrained, multi-unit resources. By designing an auction to include information about agents' uncertainty over resource use, our method can effectively allocate resources to the full extent available, while at the same time limiting resource violations. We demonstrated an efficient implementation for ACCR with an MILP. We also presented an algorithm for individual agents that uses off-the-shelf tools to generate bids. We discussed how ACCR can be applied to non-cooperative scenarios with prices. Finally, we empirically showed that ACCR outperforms other chance-constrained methods on the Maze benchmark. Future work includes extending ACCR to include broader types of resources, such as instantaneous-constrained resources, and designing custom tools to generate bids.

Acknowledgements

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]. We would like to thank an anonymous reviewer for their input on the exact MILP formulation in Proposition 1.

References

Agrawal, P.; Varakantham, P.; and Yeoh, W. 2016. Scalable Greedy Algorithms for Task/Resource Constrained Multi-Agent Stochastic Planning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*.

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

Amir, O.; Sharon, G.; and Stern, R. 2015. Multi-Agent Pathfinding as a Combinatorial Auction. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Ayton, B. J.; and Williams, B. C. 2018. Vulcan: a Monte Carlo Algorithm for Large Chance Constrained MDPs with Risk Bounding Functions. In *Computing Research Repository*.

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

Boutilier, C.; and Lu, T. 2016. Budget Allocation using Weakly Coupled, Constrained Markov Decision Processes. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*.

Capitan, J.; Spaan, M. T.; Merino, L.; and Ollero, A. 2013. Decentralized Multi-Robot Cooperation with Auctioned POMDPs. *Journal of Artificial Intelligence Research*, 32(6): 650–671.

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

de Nijs, F.; Walraven, E.; de Weerdt, M.; and Spaan, M. 2017. Bounding the Probability of Resource Constraint Violations in Multi-Agent MDPs. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*.

de Nijs, F.; Walraven, E.; De Weerdt, M.; and Spaan, M. 2021. Constrained Multiagent Markov Decision Processes: A Taxonomy of Problems and Algorithms. *Journal of Artificial Intelligence Research*, 70: 955–1001.

Dobzinski, S.; and Nisan, N. 2010. Mechanisms for Multi-Unit Auctions. *Journal of Artificial Intelligence Research*, 37: 85–98.

Dolgov, D. A.; and Durfee, E. H. 2006. Resource Allocation among Agents with MDP-Induced Preferences. *Journal of Artificial Intelligence Research*, 27: 505–549.

Forejt, V.; Kwiatkowska, M.; and Parker, D. 2012. Pareto Curves for Probabilistic Model Checking. In *Proceedings of the Tenth International Conference on Automated Technology for Verification and Analysis*.

Gautier, A.; Stephens, A.; Lacerda, B.; Hawes, N.; and Wooldridge, M. 2022. Negotiated Path Planning for Non-Cooperative Multi-Robot Systems. In *Proceedings of the Twenty-First International Conference on Autonomous Agents and Multiagent Systems*.

Gerkey, B. P.; and Mataric, M. J. 2002. Sold!: Auction Methods for Multirobot Coordination. *IEEE Transactions on Robotics and Automation*, 18(5): 758–768.

Giuseppi, A.; and Pietrabissa, A. 2020. Chance-Constrained Control With Lexicographic Deep Reinforcement Learning. *IEEE Control Systems Letters*, 4(3): 755–760.

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

Haskell, W. B.; and Jain, R. 2015. A Convex Analytic Approach to Risk-Aware Markov Decision Processes. *SIAM Journal on Control and Optimization*, 53(3): 1569–1598.

Hunsberger, L.; and Grosz, B. J. 2000. A Combinatorial Auction for Collaborative Planning. In *Proceedings of the Fourth International Conference on MultiAgent Systems*.

Koenig, S.; Tovey, C.; Lagoudakis, M.; Markakis, V.; Kempe, D.; Keskinocak, P.; Kleywegt, A.; Meyerson, A.; and Jain, S. 2006. The Power of Sequential Single-Item Auctions for Agent Coordination. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*.

Kwiatkowska, M.; Norman, G.; and Parker, D. 2002. PRISM: Probabilistic Symbolic Model Checker. In *In Proceedings of the Twelfth International Conference on Computer Performance Evaluation, Modelling Techniques and Tools*.

L. A., P.; and Fu, M. C. 2022. Risk-Sensitive Reinforcement Learning via Policy Gradient Search. *Foundations and Trends Machine Learning*, 15(5): 537–693.

Lagoudakis, M. G.; Markakis, E.; Kempe, D.; Keskinocak, P.; Kleywegt, A. J.; Koenig, S.; Tovey, C. A.; Meyerson, A.; and Jain, S. 2005. Auction-Based Multi-Robot Routing. In *Proceedings of the First International Conference on Robotics: Science and Systems*.

Lehmann, D.; Müller, R.; and Sandholm, T. 2006. The Winner Determination Problem. *Combinatorial Auctions*, 297–318.

Meuleau, N.; Hauskrecht, M.; Kim, K.-E.; Peshkin, L.; Kaelbling, L. P.; Dean, T. L.; and Boutilier, C. 1998. Solving Very Large Weakly Coupled Markov Decision Processes. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*.

Roijers, D. M.; Vamplew, P.; Whiteson, S.; and Dazeley, R. 2013. A Survey of Multi-Objective Sequential Decision-Making. *Journal of Artificial Intelligence Research*, 48: 67–113.

Santana, P.; Thiébaux, S.; and Williams, B. 2016. RAO*: An Algorithm for Chance-Constrained POMDP's. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*.

Schillinger, P.; Bürger, M.; and Dimarogonas, D. V. 2018. Auctioning over Probabilistic Options for Temporal Logic-Based Multi-Robot Cooperation under Uncertainty. In *Proceedings of the IEEE International Conference on Robotics and Automation*.

Vickrey, W. 1961. Counterspeculation, Auctions, and Competitive Sealed Tenders. *The Journal of Finance*, 16(1): 8–37.

Walraven, E.; and Spaan, M. T. 2018. Column Generation Algorithms for Constrained POMDPs. *Journal of Artificial Intelligence Research*, 62: 489–533.

Wu, J.; and Durfee, E. H. 2010. Resource-Driven Mission-Phasing Techniques for Constrained Agents in Stochastic Environments. *Journal of Artificial Intelligence Research*, 38: 415–473.

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