Risk-Aware Conditional Replanning for Globally Constrained Multi-Agent Sequential Decision Making

Frits de Nijs
Monash University
Australia
frits.nijs@monash.edu

Peter J. Stuckey
Monash University
Australia
peter.stuckey@monash.edu

ABSTRACT

Collaborating agents typically must share limited resources, such as power or bandwidth. When dealing with global constraints on resource use, agents need to plan their decisions in advance to maximize utility obtained from resources. However, deciding which agent should claim a resource under uncertainty is a hard problem: we prove that optimally planning for a globally constrained, multi-agent Markov decision process is PSPACE-hard, even when agents' transition and reward dynamics are independent, resource consumption is binary, and only one constraint is active for any decision.

To overcome this complexity, relaxations may be used to find high-value policies efficiently. Unfortunately, relaxed policies are not guaranteed to satisfy the constraints in every realizable trajectory, making them unusable in practice. In this paper, we address this weakness by investigating the use of such efficient-but-unsafe algorithms in online replanning. We show that replanning can be used to obtain high-quality safe solutions, by replanning *conditionally* with a Lagrangian relaxation-based column generation procedure. By replanning only when the risk of constraint violations becomes too high, both the computational cost and the obtained value can be improved over naive replanning, while retaining safety with respect to the constraints.

ACM Reference Format:
Frits de Nijs and Peter J. Stuckey. 2020. Risk-Aware Conditional Replanning for Globally Constrained Multi-Agent Sequential Decision Making. In *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), Auckland, New Zealand, May 9–13, 2020,* IFAAMAS, 9 pages.

1 INTRODUCTION

When multiple independent agents must share limited resources, such as power or bandwidth, we want them to plan their decisions in advance to maximize utility subject to the resource limits. Unfortunately the shared resource means that each agents plans cannot be made in isolation. A straightforward approach to this constrained multi-agent decision making problem simply joins the planning of the agents together into a single large agent and plans there actions jointly. While this approach makes it easy to ensure that resource limits are respected, it quickly becomes intractable as the resulting state space is exponential in the number of agents.

Another approach to the problem is to relax the problem and have the agents respect the resource limits *in expectation* rather than strictly. While this makes the resolution of the problem much more feasible, it will create violations of the resource constraints in practice, when the actual actions chosen by the agents overload the resource. In many cases violation of the constraints is either impossible, for example, if all customers of an electricity network attempt to use more than the networks available power then brown-outs or black-outs will occur; or dangerous, if individual agents on an autonomous robot simultaneously demand more power than the rating of the battery a fire may result. And in the case of a traffic routing optimization that promises a lower travel time by occasionally recommending detours, constraint violations could lead to traffic jams that erode users' trust, making them ignore the system in subsequent detour suggestions.

In this paper we investigate how to solve a globally constrained multi-agent sequential decision making problem without ever violating the constraints in a manner that is both efficient is solving time and utility of solution. Starting from a column generation approach to solving the relaxed problem we show how we can use conditional replanning to repair the policies created in order to avoid constraint violations.

The remainder of the paper is organized as follows. In the next section we formally introduce the problem of constraint sequential decision making, CMMDP. In Section 3.2 we show that the problem we tackle is PSPACE-hard by a reduction from Quantified Boolean Formula (QBF) solving. In Section 4 we introduce existing approaches to solve CMMDPs and discuss their strengths and weaknesses. Then in Section 5 we introduce our conditional replanning approach. In Section 6 we experimentally compare against alternative approaches to solving CMMDPs using two different problem domains. In Section 7 we discuss related work further, and finally in Section 8 we conclude.

2 BACKGROUND

To model sequential decision making under uncertainty, we employ the typical Markov Decision Process framework [6, 24, MDP], using the collaborative multi-agent generalization [8, MMDP]. Additionally, throughout this paper we assume that agents' transition and reward dynamics are *independent* from one another [5]. Therefore, an unconstrained instance is defined by n individual MDP models, one for each agent i:

Definition 2.1 (Independent MMDPs). A multi-agent MDP consisting of n independent agents can be defined through the tuple $\langle h, \{S_i, A_i, T_i, R_i\} \in \alpha\rangle$, containing

- a finite time horizon h of steps $t \in \{1, \ldots, h\}$, and
- a finite set of agents α, with each $i \in \{1, \ldots, n\}$ having
  - a finite set of local states $s_i \in S_i$,

Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), May 9–13, 2020, Auckland, New Zealand.

– a finite set of actions $a_i \in A_i$,
– a transition function $T_i : t \times S_i \times A_i \to \mathrm{P}(S_i)$, awarding probability mass $T_i(t, s_i, a_i, s_i') = \mathrm{P}(s_i' \mid t, s_i, a_i)$, and
– a reward function $R_i : t \times S_i \times A_i \to \mathbb{R}$.

The goal of the agents is to maximize their collective expected reward after h time steps. Let $\vec{S} = \prod_i S_i$ be the set of all possible joint states of the agents, and analogously $\vec{A} = \prod_i A_i$ for all joint actions. Then, the maximum expected value of the agents V^*, is defined by the Bellman equation [1957]:

$$\begin{aligned} V^*[h, \vec{s}] &= \max_{\vec{a} \in \vec{A}} R(t, \vec{s}, \vec{a}), && \forall \vec{s} \\ V^*[t, \vec{s}] &= \max_{\vec{a} \in \vec{A}} \Big(R(t, \vec{s}, \vec{a}) + Q^*[t, \vec{s}, \vec{a}] \Big), && \forall t, \vec{s} \\ Q^*[t, \vec{s}, \vec{a}] &= \sum_{\vec{s}' \in S} \Big(T(t, \vec{s}, \vec{a}, \vec{s}') \cdot V^*[t+1, \vec{s}'] \Big). && \forall t, \vec{s}, \vec{a} \end{aligned} \tag{1}$$

An optimal joint policy $\pi^* : t \times \vec{S} \to \vec{A}$ prescribes the action to take in each time and state, i.e., the $\arg\max_{\vec{a} \in \vec{A}}$ that achieves V^* in Equation (1). The task of a planning algorithm is to find π^*.

In the case of fully independent agents, the problem of finding policy π^* decomposes into planning n individual policies π_i^*. However, this changes when constraints are added to the problem.

3 CONSTRAINED SEQUENTIAL DECISION MAKING

A *constraint* imposes a restriction on the actions that the agents are allowed to execute. For example, when routing traffic to minimize total travel time, the number of vehicles taking a particular road at any given time is limited by the capacity of the road, i.e., a 3-lane road segment will fit at most 3 cars side-by-side. To model such a problem, we extend Independent MMDPs by imposing an upper limit L_t on the consumption, and adding for every agent i, a binary resource consumption function C_i:

Definition 3.1 (Constrained MMDPs). A multi-agent MDP with constraints is defined by tuple $\langle h, L, \{S_i, A_i, T_i, R_i, C_i\} \in \alpha \rangle$, consisting of

- an Independent MMDP $\langle h, \{S_i, A_i, T_i, R_i\} \in \alpha \rangle$ (Def. 2.1),
- a set L containing t real-valued limits $L_t \in \mathbb{R}_+$, and
- a binary consumption function $C_i : t \times S_i \times A_i \to \{0, 1\}$.

The constraints imposed by a Constrained MMDP (CMMDP) are satisfied by a (joint) policy π, if it holds that

$$\forall t, \vec{s}: \ \sum_{i=1}^{n} C_i(t, \vec{s}_i, \pi_i(t, \vec{s})) \leq L_t. \tag{2}$$

A policy that satisfies the constraints is *safe*. The goal of a planner for CMMDPs is to obtain the optimal safe policy π^*, which maximizes the expected reward without exceeding the capacity limits on any of the t constraints in any possible state $\vec{s}$. First, we define the set of safe joint actions in $\langle t, \vec{s} \rangle$ as

$$\vec{A}^{\text{safe}}_{t,\vec{s}} = \Big\{ \vec{a} \ \Big| \ \textstyle\sum_{i=1}^{n} C_i(t, \vec{s}_i, \vec{a}_i) \leq L_t, \vec{a} \in \vec{A} \Big\}. \tag{3}$$

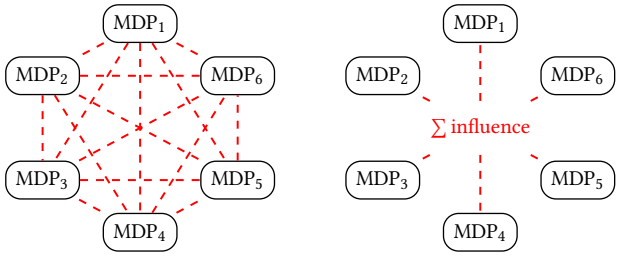

Figure 1: Constraints impose an all-to-all coupling between the otherwise independent single-agent MDPs (*left*). The weak coupling imposed by constraints suggests abstracting the influence agents have on each other (*right*). Figure reproduced with permission from [9].

Then, the optimal safe value function V^*_{safe} is given by

$$\begin{aligned} V^*_{\text{safe}}[h, \vec{s}] &= \max_{\vec{a} \in \vec{A}^{\text{safe}}_{t,\vec{s}}} R(t, \vec{s}, \vec{a}), && \forall \vec{s} \\ V^*_{\text{safe}}[t, \vec{s}] &= \max_{\vec{a} \in \vec{A}^{\text{safe}}_{t,\vec{s}}} \Big(R(t, \vec{s}, \vec{a}) + Q^*_{\text{safe}}[t, \vec{s}, \vec{a}] \Big), && \forall t, \vec{s} \\ Q^*_{\text{safe}}[t, \vec{s}, \vec{a}] &= \sum_{\vec{s}' \in S} \Big(T(t, \vec{s}, \vec{a}, \vec{s}') \cdot V^*_{\text{safe}}[t+1, \vec{s}'] \Big). && \forall t, \vec{s}, \vec{a} \in \vec{A}^{\text{safe}}_{t,\vec{s}} \end{aligned}$$

Without loss of generality, we assume no dead-ends, states where no further actions can safely be taken, exist in a Constrained MMDP. To remove dead-ends from a model, we can always add a dummy feasibility action with 0 consumption and negative infinity rewards.

3.1 Connections with other constrained models

Several models for MDPs with one primary objective and multiple secondary constraints exist, starting from at least the works of Kallenberg [18], Rossman [26] and Beutler and Ross [7]. Broadly speaking, constraints fall in two categories: soft constraints, which apply to the expected value of consumption, and hard constraints which require that even in the worst-case trajectory the constraints should be met. Tractable algorithms for soft constraints based on linear programming allow us to find solutions for these problems efficiently, see [3] for an overview.

However, hard constraints, such as those of the CMMDP model considered here, can be significantly harder to solve, especially when multiple agents are involved. This is because, while the reward function only affects the goals and behavior of one individual agent, constraints set the operating conditions for all agents together, thereby fully coupling them. Despite this, the 'independent agent core' suggests that these problems are only weakly coupled [2, 21]. As the total resource consumption is the sum of individual agent consumptions, for the sake of meeting the constraint it does not matter which specific agents contributed to the total. From the perspective of a specific agent, the influence of the other agents is thus *anonymous* [25, 28]. Intuitively, tractability may be achievable, by approximating the influence agents exert on one another, see Figure 1.

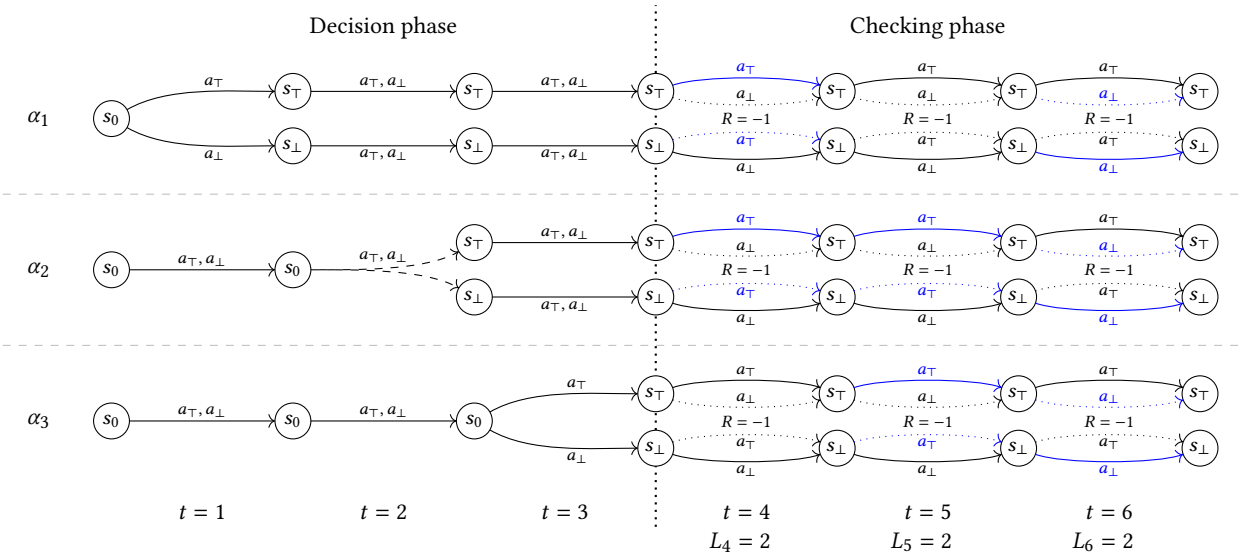

Figure 2: CMMDP corresponding to QBF (4). Transitions labeled with the corresponding action(s), solid and dotted edges are deterministic, dashed edges have $P = 0.5$. During the checking phase, blue edges have $C = 0$, while dotted edges score $R = -1$.

3.2 Complexity of planning for CMMDPs

Although several heuristic and relaxation algorithms have been proposed for (models closely related to) Constrained MMDPs, the complexity of these problems has not yet been formally established. Our first contribution is proving that even under our restricted constraint model, computing an optimal safe policy is PSPACE-hard. We do so through a reduction from the Quantified Boolean Formula (QBF) problem, which asks us to decide if a Boolean satisfiability problem augmented with existential and universal quantifiers can always be made true. The QBF problem is known to be PSPACE-complete [17, section 11.3.4]. We first present the QBF problem formally, before presenting our proof.

Definition 3.2 (QBF). A Quantified Boolean Formula ϕ in *prenex conjunctive normal form (PCNF)* is of the form $Q_1 x_1 \dots Q_n x_n \psi$, with quantifier $Q_i \in \{\forall, \exists\}$ and ψ a Boolean satisfiability formula in conjunctive normal form.

Any QBF can be converted into an equivalent QBF in PCNF through a suitable formula translation scheme. As an example of a QBF, consider the following formula:

$$\exists x_1 : \forall x_2 : \exists x_3 : (x_1 \vee x_2) \wedge (x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \quad (4)$$

The QBF problem asks to decide if this is a true formula, i.e., if there exists an allocation to x_1 *such that* for all allocations to x_2 there exists an allocation to x_3 that makes all clauses true. This example formula is true, because we can set x_1 to true to satisfy the first clause, after which we only need to set $x_3 = \neg x_2$, to cover the remaining of the two clauses that is not satisfied by either assignment to x_2.

In the subsequent proof, we show that any QBF problem can be encoded as an equivalent CMMDP in polynomial time. Intuitively, the sequential nature of truth assignments to variables, where the decision on x_1 needs to be made before x_2 is known maps naturally to the sequential decision making process of MDPs. We make use of this idea in the reduction by deciding on each variable in turn in a decision phase. The decision phase is followed by a clause checking phase, where one clause is checked every time step, using constraints to force each clause to be satisfied. When a clause cannot be satisfied by the truth assignment, the agents incur a reward penalty; only when a policy can be found that never incurs penalties is the QBF true.

Theorem 3.3. *Solving a CMMDP is PSPACE-hard.*

Proof. Given a QBF in PCNF, where the variables are numbered in order of quantification we construct the following CMMDP: create one MDP for each variable x_i. The MDP i starts in an initial state s_0, and there are two other states s_{true} or s_{false}. Each agent has two actions a_{true} and a_{false}. These actions initially transition from s_0 to s_0, until the time that $t = i$. How each of the other states is reached at $t = i$ depends on the quantifier attached to x_i. In the case of $\forall x_i$, *both* actions have a probability 0.5 of reaching either s_{true} or s_{false}, e.g., $T(i, s_0, \cdot, s_{\text{true}}) = 0.5$. In the case of $\exists p_i$, the agent can choose a_{true} to transition to s_{true} with certainty, or a_{false} to instead visit the other state. After transitioning from s_0 the agents will follow one of two self-transitions (e.g., $s_{\text{true}} \to s_{\text{true}}$). This first *decision* phase lasts until $t = n + 1$, at which point all agents have transitioned to either s_{true} or s_{false}, forming a candidate truth assignment. During the decision phase, agents neither receive rewards, nor consume resources, i.e., $\forall t \leq n$: $R_i(t, \cdot, \cdot) = C_i(t, \cdot, \cdot) = 0$.

Following the decision phase there are $|\psi|$ clause checking time steps. During the clause checking phase, agents continue to follow the self-transitions, but now they are penalized for choosing the action that disagrees with their state, $\forall t > n$: $R_i(t, s_{\text{true}}, a_{\text{false}}) = R_i(t, s_{\text{false}}, a_{\text{true}}) = -1$, and 0 otherwise. Furthermore, agents now consume 1 resource *unless* the action they choose satisfies the corresponding clause, i.e., $\forall t > n$: $R_i(t, \cdot, a_{\text{false}}) = 0$ iff $\neg x_i \in \psi_{t-n}$. The constraint L_t in time $t > n$ imposes a maximum consumption

Algorithm 1 Constrained MDP Occupancy Linear Program [3].

Input: CMMDP $\langle h, L, \{S_i, A_i, T_i, R_i, C_i\} \in \alpha \rangle$
Output: n independent stochastic policies π_i, see (5)

$$\max_{x_{i,t,s,a}} \sum_{i=1}^{n} \sum_{t=1}^{h} \sum_{s \in S_i} \sum_{a \in A_i} R_i(t,s,a) \cdot x_{i,t,s,a}$$

$$\text{s.t.} \sum_{a' \in A_i} x_{i,t+1,s',a'} = \sum_{s \in S_i, a \in A_i} T_i(t,s,a,s') \cdot x_{i,t,s,a} \qquad \forall i, t < h, s'$$

$$\sum_{a' \in A_i} x_{i,1,s',a'} = \mathrm{P}_i(1,s') \qquad \forall i, s'$$

$$\sum_{i, s \in S_i, a \in A_i} C_i(t,s,a) \cdot x_{i,t,s,a} \leq L_t \qquad \forall t$$

$$0 \leq x_{i,t,s,a} \leq 1$$

of $n-1$, forcing at least one of the actions chosen to make the clause true. However, unless all chosen actions align with their states (= assignments), satisfying the constraint incurs a penalty of at least -1. Therefore, the optimal policy for this CMMDP has an expected value $V^*[1, \vec{s}_0] = 0$ only if the corresponding QBF is true.

The size of the resulting CMMDP is polynomial in the size of the input QBF: there are n agents, each with 3 states and 2 actions, and a planning horizon $h = n + |\psi|$. □

Figure 2 applies the reduction in the proof to our example QBF formula from Equation (4). As a corollary to this proof, observe that an agent i that decides on an existential quantifier at time t needs to take into account the state transitions of all the agents that participate in one of the clauses with i and already transitioned to s_{true} or s_{false} at an earlier time $t' < t$. This means that an optimal policy for a CMMDP necessarily needs to consider the joint state of all agents in the worst case, even though the agents' transition and reward models are otherwise independent.

4 EXISTING APPROACHES

Because of the high complexity of solving CMMDPs, finding optimal policies is intractable in general. Existing approximate solutions based on principled optimization therefore either tighten or relax aspects of the problem to retain tractability. In this section, we describe existing methods with optimality guarantees (within their specific assumptions) for both categories.

4.1 Constraint relaxations

To obtain a policy efficiently, we either have to settle for approximate solutions, or forgo constraint strictness by accepting *relaxed* solutions. Such solutions typically satisfy the constraints in expectation, guaranteeing only that $\mathbb{E}[\sum_i C_i] \leq L_t$. Several algorithms have been proposed to compute relaxed solutions, here we briefly describe four possible approaches.

Constrained MDP dual LP. One approach to compute relaxed solutions is to solve the linear program (LP) given in Algorithm 1, which is the extension of a Constrained MDP [3] to multiple agents. This LP is defined over *probability flow* variables $x_{i,t,s,a}$, which represent the unconditional probability that agent i takes action a at time t in state s. The flow of probability is made consistent with the transition function through the first two constraints, starting from prior probabilities $\mathrm{P}_i(1,s')$ over the initial state. The resource usage constraints imposed by the CMMDP are satisfied in expectation through the third constraint: because $x_{i,t,s,a}$ is the marginal probability of incurring resource cost $C_{i,j}(t,s,a)$, the sum over all possible states gives us the *expected* resource consumption, thus making the policy satisfy the constraints in expectation. The policy indicated by the flow variables is stochastic, with actions chosen according to the probability

$$\Pr(\pi_i(t,s_i) = a_i) = \frac{x_{i,t,s_i,a_i}}{\sum_{a_j} x_{i,t,s_i,a_j}}. \tag{5}$$

i-dual. The LP at the core of Algorithm 1 can quickly grow very large, having $O(nh|S_i||A_i|)$ variables. However, many of the variables may never be assigned a positive probability value $x > 0$. Trevizan et al. [27] use this insight to speed up solving the LP, by generating variables incrementally, as they receive positive probability. It does this by creating artificial goal variables $\hat{x}_{i,t,s}$ for the states at the reachable fringe, which are only expanded into their actual variables $x_{i,t,s,a}$ when a previous iteration indicated $\hat{x}_{i,t,s} > 0$. To guide the expansion of promising candidates $\hat{x}_{i,t,s}$, the variables are assigned (admissible) heuristic values for both the expected future reward and expected future costs. In the case of our CMMDP problems, we use the admissible heuristic value given by computing the optimal unconstrained single-agent policies' $V_i[t,s_i]$, and setting consumption to 0.

Lagrangian Relaxation and Column Generation. Especially when dealing with multiple independent agents, an efficient approach to obtain relaxed policies is to make use of Lagrangian relaxation [15]. Yost and Washburn [31] propose a column generation approach based on Lagrangian relaxation, interleaving single-agent (partially observable) MDP planning for a resource usage cost with solving a 'master' linear program that finds the best updated resource price.

Column Generation [14] allows us to decompose combinatorial optimization problems, provided the problem has some method to generate new potential solutions efficiently. The technique uses the insight that, when an LP is used to select solutions from an exhaustive set Z, the simplex algorithm only considers one variable at a time. If we can generate the optimal element to be selected on the fly, we avoid having to maintain the exhaustive set of candidate variables explicitly. In the case of CMMDPs, the elements $\langle i,k \rangle$ of the set Z are the individual agent MDP policies, with expected value $V_{\pi_{i,k}}$ and expected resource usage $U_{\pi_{i,k}}$. The master LP selects the optimal 'mix' of policies, by awarding a probability $x_{i,k}$ to each policy $\pi_{i,k}$ such that the expected value is maximized within the constraints:

$$\begin{aligned} \max_{x_{i,k}} & \sum_{i=1}^{n} \sum_{\pi_k \in Z_i} x_{i,k} V_{\pi_{i,k}}(1,s_1), \\ \text{s.t.} & \sum_{i=1}^{n} \sum_{\pi_k \in Z_i} x_{i,k}\, U_{\pi_{i,k},j}(1,s_1) \leq L_j \qquad \forall j, \\ & \sum_{\pi_k \in Z_i} x_{i,k} = 1, \qquad \forall i, \\ & x_{i,k} \geq 0, \qquad \forall i,k. \end{aligned} \tag{6}$$

Algorithm 2 Column generation for CMMDPs [31].

Input: CMMDP $M = \langle h, L, \{S_i, A_i, T_i, R_i, C_i\} \in \alpha \rangle$
Output: Deterministic policies $\pi_{i,k} \in Z$, with $\Pr(\pi_{i,k}) = x_{i,k}$

$\lambda = 0, \lambda' = \infty, \Lambda = \emptyset, Z = \emptyset$
while $\lambda \neq \lambda'$ **do**
  $\lambda \leftarrow \lambda'$
  $\forall i \colon \pi_{i,\text{new}} \leftarrow \textsc{plan}(M_i, \lambda)$ ▷ Equation (7)
  $Z_i \leftarrow Z_i \cup \pi_{i,\text{new}}$
  $\langle x, \lambda' \rangle \leftarrow \textsc{solveLP}(Z)$ ▷ Equation (6)
  $\Lambda \leftarrow \Lambda \cup \lambda$
end while
return $\langle x, Z, \Lambda \rangle$

Generating the 'column' of constants $V_{\pi_{i,k}}, U_{\pi_{i,k}}$ that define the candidate variable comes down to planning a single-agent MDP policy for a resource usage cost λ, for example through a modified dynamic programming, for equation

$$V_{\pi_{i,k}}(t,s) = \max_{a \in A} R_i(s,a) + \sum_{s' \in S_i} T_i(s,a,s') \cdot V_{\pi_{i,k}} - \lambda_k \times C_i(t,s,a). \tag{7}$$

The resulting algorithm 2 finds the optimal stochastic mix of policies for relaxed constraints, satisfying the constraints in expectation, in worst-case polynomial time. Because the resulting policy is a stochastic mix, every time we want to execute the joint policy, we first have to sample a $\pi_i \in Z \propto x_i \ \forall i$.

Dynamic bounding. All three previous algorithms satisfy the constraints in expectation. However, even when constraints are soft, exceeding them typically incurs some cost to the system operator. Therefore, De Nijs et al. [11] propose to restrict the probability that the random variable X_t of the total resource consumption exceeds the limit, through Hoeffding's inequality [16] on the tail probability of a sum of independent random variables $X_{i,t}$. This allows us to find the maximum L_t^* for which

$$\Pr\left[X_t > L_t \mid \mathbb{E}\left[X_t\right] \leq L_t^*\right] \leq \alpha, \quad \forall t.$$

These reduced constraints can subsequently be used in conjunction with any of the previously mentioned algorithms to compute α-safe policies. However, because this bound may significantly overestimate the tail probability, [11] suggests to use it as a starting point for an iterative hill-climbing procedure to relax L_t^* to tightly meet limit α.

4.2 Safe solutions and approximations

The bounding procedure gives us policies which are risk-bounded, but to obtain entirely safe policies, different algorithms are needed. The brute-force baseline approach to find safe policies is to apply dynamic programming over the joint models, at every point only selecting actions from the safe action set of Eq. (3). Unfortunately, this algorithm requires $O(2\vec{S}\vec{A}h)$ time and $O(\vec{S}h)$ space to arrive at the optimal solution in the worst case, which are both exponential in the number of agents n. A more efficient safe approximation allocates resources unconditionally, through a Mixed-Integer Linear Program (MILP):

Algorithm 3 Preallocation Mixed-integer Linear Program [30].

Input: CMMDP $\langle h, L, \{S_i, A_i, T_i, R_i, C_i\} \in \alpha \rangle$
Output: n independent stochastic policies π_i, see (5)

$$\max_{x_{i,t,s,a}} \sum_{i=1}^{n} \sum_{t=1}^{h} \sum_{s \in S_i} \sum_{a \in A_i} R_i(t,s,a) \cdot x_{i,t,s,a}$$

$$\text{s.t.} \sum_{a' \in A_i} x_{i,t+1,s',a'} = \sum_{s \in S_i, a \in A_i} T_i(t,s,a,s') \cdot x_{i,t,s,a} \qquad \forall i, t < h, s'$$

$$\sum_{a' \in A_i} x_{i,1,s',a'} = \mathrm{P}_i(1,s') \qquad \forall i, s'$$

$$\sum_{s \in S_i, a \in A_i} C_i(t,s,a) \cdot x_{i,t,s,a} \leq \Delta_{i,t} \qquad \forall i, t$$

$$\sum_{i=1}^{n} \Delta_{i,t} \leq L_t \qquad \forall t$$

$$0 \leq x_{i,t,s,a} \leq 1, \Delta_{i,t} \in \{0,1\} \qquad \forall i, t, s, a$$

Preallocation MILP. This algorithm was first proposed by Dolgov and Durfee [13] for infinite-horizon MMDPs with single-shot resource allocations, and later extended in Wu and Durfee [30] to handle resource reallocation over time. It uses the insight that the *marginal* probability of an agent arriving in a state $\langle t, s_i \rangle$ and using action a (the $x_{i,t,s,a}$ variables of Alg. 1) can be used as an indicator for an agent using a resource in the 'worst' case. By constraining the sum over these worst-case 'preallocations', we can be sure that the agents never exceed their allowance.

Algorithm 3 presents the full MILP model, using the same flow variables and constraints of Alg. 1, augmented with $\Delta_{i,j}$ variables indicating an allocation of resource t to agent i. The third constraint guarantees that any probability of taking a resource-consuming action is translated into an allocation. Finally, the fourth constraint ensures that the total allocation does not exceed the limits.

This algorithm computes the optimal resource preallocation, in worst-case time $O(2^{nm})$. This algorithm thus also requires time exponential in the number of agents, but it uses exponentially less space than dynamic programming. This saving comes at a cost, because preallocations reason over all possible paths at the same time, and may therefore over-estimate realized resource consumption, resulting in conservative policies. Trevizan et al. [27] propose a row-and-column generation procedure for probability flow LPs, which can improve the average-case runtime of the MILP.

5 CONDITIONAL REPLANNING

Unfortunately, existing algorithms are either insufficiently scalable to handle more than a handful of agents or time-steps, or relax the problem to the point that constraint satisfaction cannot be guaranteed. Therefore, we investigate the use of replanning to repair policies that are computed by relaxations.

Recall that relaxed solutions satisfy the constraints in expectation, by computing the marginal probability of realized resource consumption. Because the marginal probability of the *current* state is 1, we may expect that a relaxed solution is at least guaranteed to be feasible for the current state. However, relaxations can return

Algorithm 4 Conditional replanning for CMMDPs.

Risk threshold α, joint policy $\vec{\pi}$, warm start columns Λ, consumption sample means $\vec{\mu}'$ and standard deviations $\vec{\sigma}'$.

1: **procedure** INIT(CMMDP M, initial state $\vec{s}_0$)
2: $\langle x, Z, \Lambda \rangle \leftarrow$ PLANCG(M)
3: $\vec{\pi} \leftarrow$ SAMPLESAFE($x, Z, \vec{s}_0$)
4: $\langle \vec{\mu}', \vec{\sigma}' \rangle \leftarrow$ SAMPLECONSUMPTION($\vec{\pi}, M$)
5: **end procedure**
6: **procedure** APPLYPOLICY(time t, state $\vec{s}$)
7: **if** $C(t, \vec{s}, \vec{\pi}(\vec{s})) \geq L_t$ or
8: $\exists t' : \left| \mathbb{E}[C^{\vec{s}}_{\vec{\pi}}(t')] - \mu'_{t'} \right| \geq \alpha \cdot \sigma'_{t'}$ **then**
9: $\langle x, Z, \Lambda \rangle \leftarrow$ REPLANCG(M, Λ)
10: $\vec{\pi} \leftarrow$ SAMPLESAFE($x, Z, \vec{s}$)
11: $\langle \vec{\mu}', \vec{\sigma}' \rangle \leftarrow$ SAMPLECONSUMPTION($\vec{\pi}, M$)
12: **end if**
13: **return** action $\vec{a} = \vec{\pi}(\vec{s})$
14: **end procedure**

stochastic policies, which means that algorithms like column generation might still return policy mixes where some combinations of policies over-consume resources in the current time step. For example, a constraint of 0.6 may be satisfied by a mix where 60% policies use 1 unit and 40% policies use 0 units of the resource. Nevertheless, in the case of a single active instantaneous constraint per state, a safe policy selection is guaranteed to exist. The odds of not sampling this (current-instant) safe policy decrease exponentially, e.g., $\Pr = 0.6^k$ for k samples in the example, making it vanishingly unlikely that rejection sampling over the policy mix would not find a safe action to execute. And even if no safe policy is sampled, we can find a worst-case one by construction, by selecting for each agent the policy with least immediate consumption. One approach to address the issue of constraint violations is thus to replan a control policy after every time step using a relaxing solver, and using rejection sampling to guarantee safe action selection.

Warm replanning in column generation. Naive replanning is costly if done from scratch, and usually cannot be done under time constraints from policy execution [12]. Fortunately, in the case of column generation we have the opportunity to reuse information from the previous time step, for a warm restart. After a state transition the expected value $V_{\pi_{i,k}}(t, s)$ and expected resource usage $U_{\pi_{i,k}}(t, s)$ constants that make up the (surviving) columns in Z are no longer valid, because they are specific to the initial state (t, s). However, the policy $\pi_{i,k}$ remains a valid policy for costs λ_k, which means that we can compute updated column values $V_{\pi_{i,k}}(t{+}1, s')$ and $U_{\pi_{i,k}}(t{+}1, s')$ without having to recompute the policy. This allows us to start the next iteration of column generation with the same set of lambda costs that were active previously.

Minimizing the frequency of replanning. While warm restarts help to bring down the computational impact of replanning, it remains costly to replan every time step, and multi-agent settings have the added challenge that a replanning operation also incurs a communication overhead [29]. Furthermore, because finding a safe action requires rejection sampling, the chosen policy is always on the conservative half of the expected value, making the actually implemented policies considerably more conservative than the expectation. By resampling unconditionally, every time step incurs this reward penalty. What is needed therefore is a mechanism to optimize the frequency of replanning [1].

We propose the following replanning strategy, presented formally in Algorithm 4: given a selection of policies $\vec{\pi} = \{\pi_1, \ldots \pi_n\}$ sampled to implement a safe action in some previous time step t', we only replan a new mix of policies at time t if either one of two conditions hold: (line 7) the currently chosen action would immediately violate the currently active constraints, or (line 8) the (estimated) future probability of a constraint violation using the current policy exceeds a given risk threshold α. This forward-looking perspective is necessary for models such as the QBF mapping presented in Section 3.2, where the decisions that impact resource consumption are made several time steps before a constraint applies.

In order to obtain the true future probability of a constraint violation, we would need to join the individual agent models together and find the marginal probability of all joint states where a given constraint is violated. However, because the joint model has exponential size and is therefore too large to evaluate, we instead use a simple forward simulations of the policies to estimate the risk of a future constraint violation.

In case replanning is required, we use the same rejection sampling technique to obtain a new selection $\vec{\pi}$. The effect of this replanning strategy is that the currently selected policy is retained, unless stochastic transitions push the policy into an unsafe continuation.

6 EMPIRICAL EVALUATION

In order to evaluate conditional replanning empirically, we compare it against the existing baseline algorithms presented in Section 4. We compare our results on a challenging smart energy domain involving the allocation of power to HVAC units under a time-fluctuating renewable power production.

Planetary Rover (Maze). The Mars rover domain proposed by Wu and Durfee [30] models a collection of autonomous vehicles exploring remote locations. Each of the vehicles operates in its own grid-world, in which it has to perform as many tasks as possible before the vehicle expires. In this domain the resources allow the agent to traverse the world more safely, while every task *requires* a resource to complete.

The grid-world of a vehicle consists of $m \times m$ cells, which represent the states that the vehicle can be in. Only a subset of the cells are traversable, the other cells act as walls. Traversable cells may additionally have a task to be completed, which is known to the agents at plan time. Beyond the locations in the grid, agents also have an 'expired' state, which is reached when the vehicle breaks down. The probability that an agent breaks down depends on the type of 'move' it uses. A *safe* move requires a resource to proceed, but gives only 5% chance of breaking down, whereas an *unsafe* move breaks the vehicle with 20% chance. For the resource model, we use the model where the agents are allowed to change their resource consumption in every time step. This means that resources are effectively renewable, and corresponds to the unlimited phase shifting model of Wu and Durfee [30].

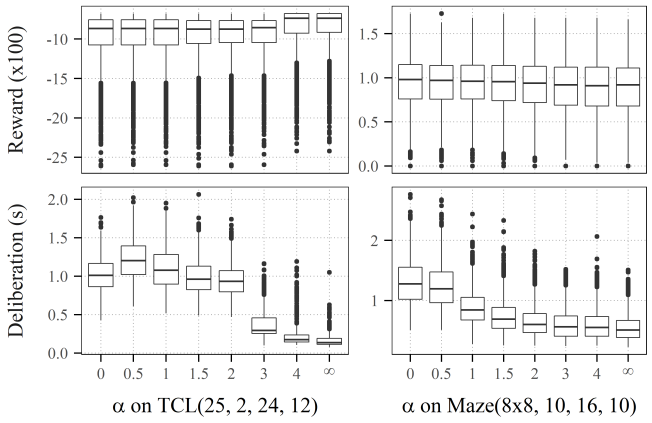

Figure 3: Effect of conditional replanning parameter α on algorithm performance. Instance size tuple is $(|S|, |A|, h, n)$.

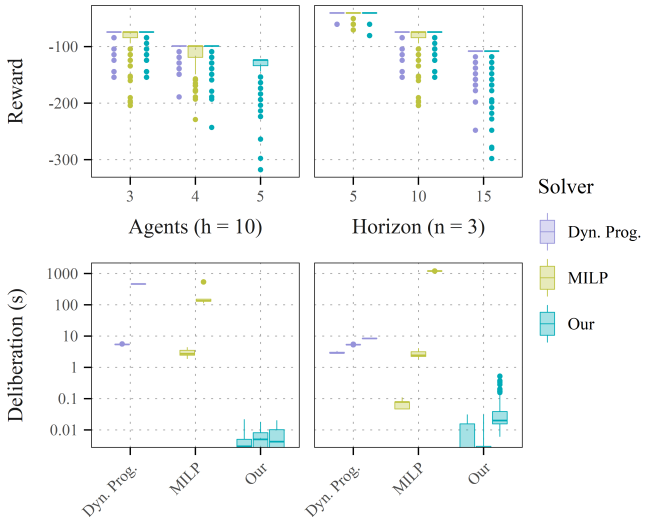

Figure 4: Scalability limits of exhaustive solvers compared to replanning on small instances of the TCL problem.

Thermostatically Controlled Loads (TCL). In this problem described by De Nijs et al. [10], each of the n agents is responsible for keeping the temperature inside its house at a level that is comfortable to its occupants (i.e., as close as possible to the given set point). The controller has two actions available to it, which are either to heat, or do nothing. Heating costs power at one unit (meaning all the houses use an equivalent amount of power). However, the houses under control each have different build quality, which affects the rate at which the building heats up and cools down, according to a first-order exponential decay function of a controlled temperature process described by Mortensen and Haggerty [22]. The continuous temperature values are discretized into temperature ranges, with the transition function mapping the probability that a value in range s would advance to range s' according to its continuous transition. The power limit that the agents are subject to is a periodic waveform on top of baseline production, modeling the influence of solar power generation.

6.1 Results

We first perform a preliminary experiment to evaluate the impact of different settings of risk tolerance α, performing 250 repetitions across 10 small instances. We record *per-run* rewards, as well as the deliberation time (initial planning time plus time spent replanning). Figure 3 presents the results, demonstrating a strong reduction in runtime as α increases. The impact of α on reward is relatively minor on our domains, because our domains do not have as significant a delay between decisions and resource consumption as in QBF. The direction of the reward trend is specific to the domain dynamics. On TCL, always replanning is too conservative, resulting in lower rewards. As a result, $\alpha = \infty$ dominates the other choices. On the Maze domain, replanning more often increases reward, which we hypothesize may be helped by a domain-specific effect: rovers can break every time step, therefore adjusting the resource allocation from dead to live agents improves the average performance. We set $\alpha = 1.5$ on Maze, to balance between runtime benefits and reward.

Next we compare our replanning approaches against the existing safe approaches with optimality guarantees, in order to evaluate their scalability limits and how much performance is lost by repairing a relaxed solution through rejection sampling and replanning. In this first experiment, we focus on the TCL domain because it has a much smaller action space than Maze, making it feasible to evaluate dynamic programming over the joint models. We randomly generate 5 instances and evaluate each solver 50 times, recording for each evaluation: the obtained reward and the total deliberation time required to plan and execute the policy, with a timeout of 20 minutes. Figure 4 shows the performance trends of the baseline algorithms dynamic programming and mixed-integer linear programming, compared with conditional replanning.

We observe that finding the optimal policy (through dynamic programming) or the optimal resource preallocation (using the MILP) quickly becomes intractable as the number of agents or the horizon increases. Because the optimal resource preallocation is conservative under uncertainty, the reward obtained by MILP is slightly lower than that of dynamic programming. In contrast, our conditional replanning approach is able to effectively redistribute resources when needed, obtaining very close to optimal reward while taking only a fraction of the runtime.

Finally, we investigate the scalability of replanning on long horizons and for large agent counts, beyond sizes that can be handled by either dynamic programming or MILP models. Figure 5 presents performance trends as the number of agents or planning horizons grow. We observe that, as expected, all three relaxation algorithms Constrained MDP LP (CMDP), i-dual, and Column Generation (CG) find the same solution, in terms of obtained rewards and frequency of constraint violations in policy simulation, with the only difference being the amount of time it takes to find a solution. In particular, we note that CG is sometimes able to find solutions up to an order of magnitude quicker than either CMDP or i-dual. Unexpectedly, i-dual does not provide speedup for solving the CMDP LP in our domains, even timing out on the larger Maze instances; we hypothesize that this is due to two differences in our setting. In the first place, our domains are not shortest-path problems, which

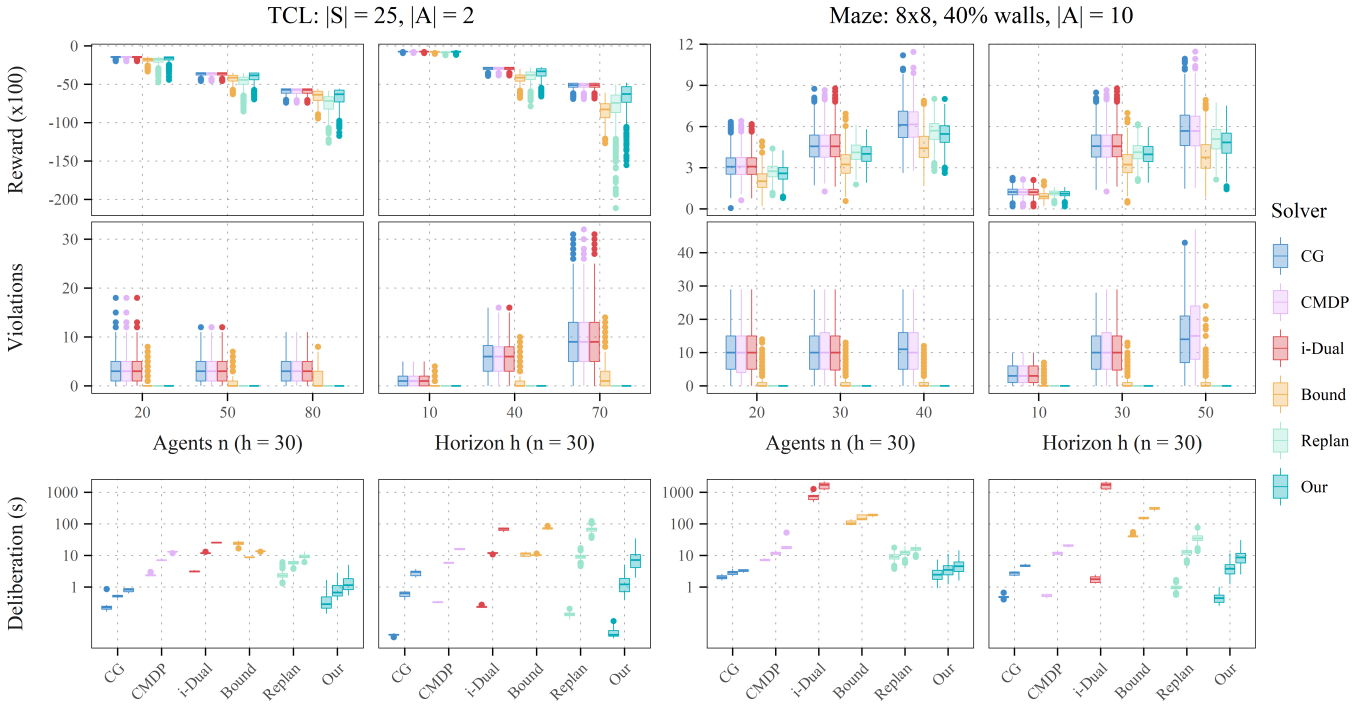

Figure 5: Solution quality, safety and scalability of conditional replanning (Our), compared to the single-shot relaxations CG, CMDP and i-Dual, and constraint-aware approaches dynamic bounding (Bound) and unconditional replanning (Replan).

means that the total state space explored is a relatively larger fraction of the total state space. Secondly, our multi-agent problems mean that the number of reachable state variables expanded with every iteration grows much quicker, as each agent has its own reachable states.

However, we further observe that policies from the relaxation algorithms regularly exceed the resource constraints, making them unsuitable for applications with hard constraints. Dynamic bounding, which we employ with maximum risk probability $\alpha = 0.05$ and the hill-climbing procedure to tightly meet this limit, obtains policies that are significantly safer while still avoiding the need for inter-agent communication during policy execution. However, this comes at the cost of both solution quality, due to the need to be conservative about resource usage, and total deliberation time, because of the need to simulate candidate solutions to obtain consumption distribution estimates for the hill-climbing step. Unconditional replanning uses inter-agent communication during policy execution to guarantee safety while improving on solution quality and total deliberation time (which includes time spent replanning) compared to dynamic bounding. Conditional replanning improves on this further, greatly reducing required deliberation time and consistently outperforming full replanning, while also further improving the cumulative reward obtained on the TCL domain.

7 RELATED WORK

Our work is closely related to the ideas of Model Predictive Control [20, MPC], which is a control paradigm where only the first step of a solution optimized over the full horizon is implemented. The full replanning scheme could be classified as an instance of MPC. MPC has previously been applied to single-agent planning problems with chance constraints [23]. The conditional replanning variant in this paper essentially the central idea of MPC, by only replanning when the previous control policy is no longer acceptable.

Another approach to handle large-scale MDPs effectively is Monte-Carlo Tree Search (MCTS) [4], which has also been applied to (single-agent) constrained MDPs [19]. An advantage of MCTS over planning is that it only has to consider states which are reachable from the current state, making it much more effective at large state-space MDPs. Nevertheless, it would be hard to apply MCTS to our multi-agent problems, as the action space is also exponentially sized, which makes the number of explored rollouts per action too small to obtain accurate expected-value estimates.

8 CONCLUSION

Globally constrained multi-agent sequential decision making problems (CMMDPs) are extremely challenging to solve. This is perhaps not too surprising since we show that they are PSPACE-hard even for a restricted class. In this paper we provide an efficient approach to solving CMMDPs that ensure that no violations of resource limits arise during execution. Our conditional replanning approach is far more efficient than the full replanning alternative, while providing solutions that obtain higher reward in practice. Our approach is comparable in efficiency to methods that solve the relaxed problem, without ever violating the constraints. In summary we provide a practically usable approach to tackling this challenging class of problems.

REFERENCES

[1] Jordan R. Abrahams, David A. Chu, Grace Diehl, Marina Knittel, Judy Lin, William Lloyd, James C. Boerkoel Jr, and Jeremy Frank. 2019. DREAM: An Algorithm for Mitigating the Overhead of Robust Rescheduling. In *Proceedings of the 29th International Conference on Automated Planning and Scheduling*.

[2] Daniel Adelman and Adam J. Mersereau. 2008. Relaxations of weakly coupled stochastic dynamic programs. *Operations Research* 56, 3 (2008), 712–727. https://doi.org/10.1287/opre.1070.0445

[3] Eitan Altman. 1999. *Constrained Markov decision processes*. Chapman & Hall/CRC.

[4] Benjamin J. Ayton and Brian C. Williams. 2018. Vulcan: A Monte Carlo Algorithm for Large Chance Constrained MDPs with Risk Bounding Functions. *CoRR* abs/1809.01220 (2018). arXiv:1809.01220

[5] Raphen Becker, Shlomo Zilberstein, Victor Lesser, and Claudia V. Goldman. 2004. Solving transition independent decentralized Markov decision processes. *Journal of Artificial Intelligence Research* 22 (2004), 423–455. https://doi.org/10.1145/860575.860583

[6] Richard E. Bellman. 1957. A Markovian Decision Process. *Journal of Mathematics and Mechanics* 6, 5 (1957), 679–684.

[7] Frederick J. Beutler and Keith W. Ross. 1985. Optimal policies for controlled Markov chains with a constraint. *J. Math. Anal. Appl.* 112, 1 (1985), 236–252. https://doi.org/10.1016/0022-247X(85)90288-4

[8] Craig Boutilier. 1996. Planning, Learning and Coordination in Multiagent Decision Processes. In *Proceedings of the 6th Conference on Theoretical Aspects of Rationality and Knowledge*. 195–210.

[9] Frits De Nijs. 2019. *Resource-constrained Multi-agent Markov Decision Processes*. Ph.D. Dissertation. Delft University of Technology.

[10] Frits De Nijs, Matthijs T. J. Spaan, and Mathijs M. De Weerdt. 2015. Best-response planning of thermostatically controlled loads under power constraints. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*. AAAI Press, 615–621.

[11] Frits De Nijs, Erwin Walraven, Mathijs M. De Weerdt, and Matthijs T. J. Spaan. 2017. Bounding the probability of resource constraint violations in multi-agent MDPs. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*. AAAI Press, 3562–3568.

[12] Thomas Dean, Leslie Pack Kaelbling, Jak Kirman, and Ann Nicholson. 1995. Planning under time constraints in stochastic domains. *Artificial Intelligence* 76, 1-2 (1995), 35–74. https://doi.org/10.1016/0004-3702(94)00086-G

[13] Dmitri A. Dolgov and Edmund H. Durfee. 2004. Optimal resource allocation and policy formulation in loosely-coupled Markov decision processes. In *Proceedings of the 14th International Conference on Automated Planning and Scheduling*. 315–324.

[14] P. C. Gilmore and R. E. Gomory. 1961. A linear programming approach to the cutting-stock problem. *Operations Research* 9, 6 (1961), 849–859.

[15] Geoffrey J. Gordon, Pradeep Varakantham, William Yeoh, Hoong Chuin Lau, Ajay S. Aravamudhan, and Shih-Fen Cheng. 2012. Lagrangian Relaxation for Large-Scale Multi-agent Planning. In *WI-IAT*. 494–501. https://doi.org/10.1109/WI-IAT.2012.252

[16] Wassily Hoeffding. 1963. Probability inequalities for sums of bounded random variables. *J. Amer. Statist. Assoc.* 58, 301 (1963), 13–30.

[17] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. 2006. *Introduction to Automata Theory, Languages, and Computation* (3 ed.). Pearson.

[18] L. C. M. Kallenberg. 1983. *Linear programming and finite Markovian control problems*. Number 148 in Mathematical Centre Tracts. Mathematisch Centrum, Amsterdam.

[19] Jongmin Lee, Geon-Hyeong Kim, Pascal Poupart, and Kee-Eung Kim. 2018. Monte-Carlo Tree Search for Constrained POMDPs. In *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.). Curran Associates, Inc., 7923–7932.

[20] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. 2000. Constrained model predictive control: Stability and optimality. *Automatica* 36, 6 (2000), 789–814. https://doi.org/10.1016/S0005-1098(99)00214-9

[21] Nicolas Meuleau, Milos Hauskrecht, Kee-eung Kim, Leonid Peshkin, Leslie Pack Kaelbling, Thomas Dean, and Craig Boutilier. 1998. Solving Very Large Weakly Coupled Markov Decision Processes. In *Proceedings of the 15th National Conference on Artificial Intelligence*. 165–172.

[22] R. E. Mortensen and K. P. Haggerty. 1988. A stochastic computer model for heating and cooling loads. *IEEE Transactions on Power Systems* 3, 3 (1988), 1213–1219.

[23] M. Ono. 2012. Joint chance-constrained model predictive control with probabilistic resolvability. In *Proceedings of the 2012 American Control Conference*. IEEE, 435–441. https://doi.org/10.1109/ACC.2012.6315201

[24] Martin L. Puterman. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc.

[25] Philipp Robbel, Frans A. Oliehoek, and Mykel J. Kochenderfer. 2016. Exploiting anonymity in approximate linear programming: scaling to large multiagent MDPs. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*. 2537–2543.

[26] Lewis A. Rossman. 1977. Reliability-constrained dynamic programing and randomized release rules in reservoir management. *Water Resources Research* 13, 2 (1977), 247–255. https://doi.org/10.1029/WR013i002p00247

[27] Felipe Trevizan, Sylvie Thiébaux, Pedro Santana, and Brian Williams. 2016. Heuristic Search in Dual Space for Constrained Stochastic Shortest Path Problems. In *Proceedings of the 26th International Conference on Automated Planning and Scheduling*. 326–334.

[28] Pradeep Varakantham, Yossiri Adulyasak, and Patrick Jaillet. 2014. Decentralized Stochastic Planning with Anonymity in Interactions. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*. AAAI Press, 2505–2511.

[29] Feng Wu, Shlomo Zilberstein, and Xiaoping Chen. 2011. Online planning for multi-agent systems with bounded communication. *Artificial Intelligence* 175 (2011), 487–511. https://doi.org/10.1016/j.artint.2010.09.008

[30] 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.

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